LA PRIMA RIVISTA EUROPEA DI SOFTWARE PER PERSONAL COMPUTER

ISSN 0392-8896

PERSONAL SOFTWARE

ANNO 4 N. 25
FEBBRAIO 1985 - L. 4.000

UNA PUBBLICAZIONE DEL GRUPPO EDITORIALE JACKSON

Spedizione in abb. postale Gruppo III/70

Copia riservata agli abbonati

APPLE:
PAGINE VIDEO E DINTORNI

ORGANO ELETTRONICO
CON LO SHARP

PHANTOMS LABIRINTO
PER VIC 20

SUPER ASSEMBLER
PER C 64

LANGUAGE TUTOR
PER SPECTRUM

SPRITE EDITOR
PER C 64

ROULETTE
CON TI99





Questo mese vi presentiamo "Il Castello", un gioco di avventura che vi appassionerà.

PERSONAL SOFTWARE
N. 25
FEBBRAIO 1985

ARTICOLI

GUIDA

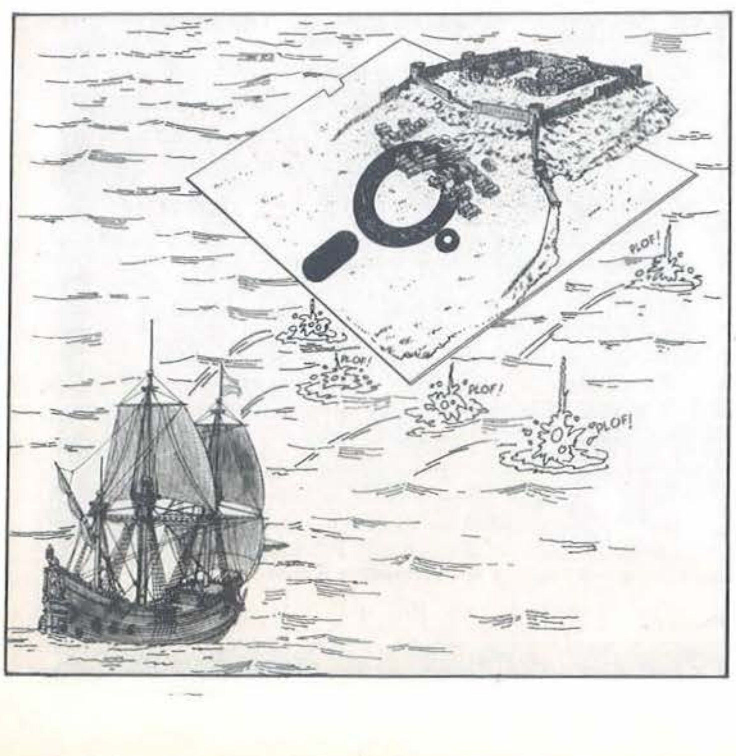
RUBRICHE

EDITORIALE

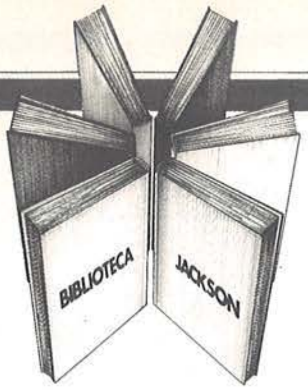
Un chilo di software

di Riccardo Paolillo

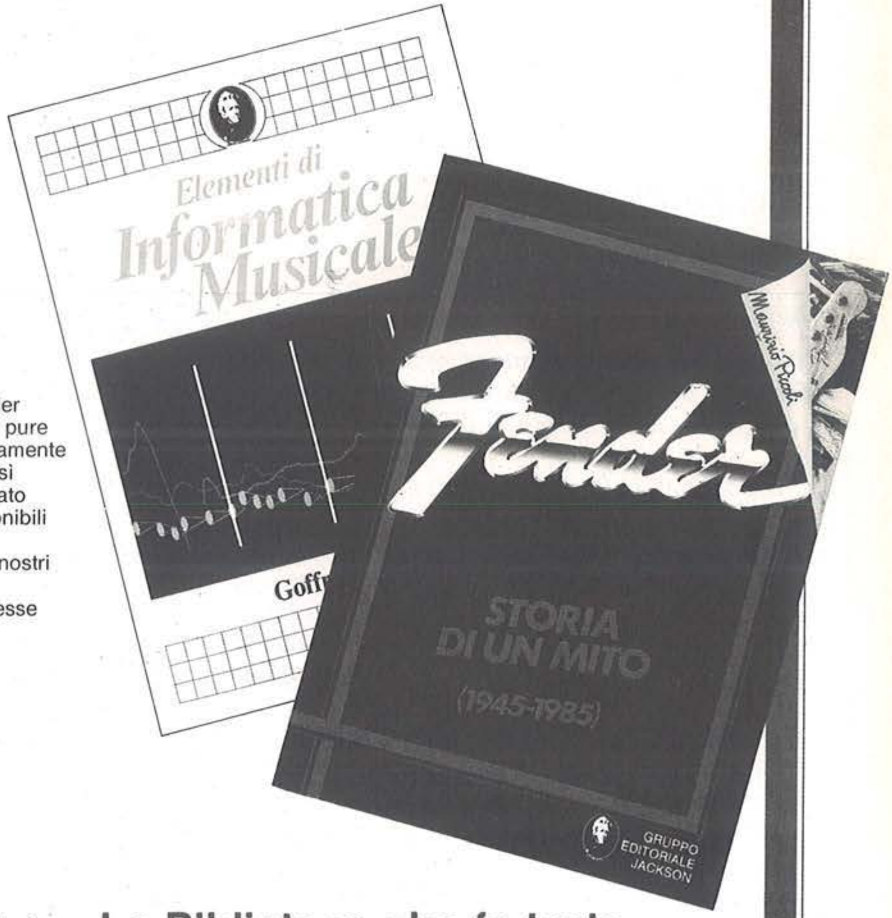
Con un titolo volutamente provocatorio, vogliamo affrontare un problema molto importante e dibattuto: la distribuzione, e quindi la diffusione, del software in Italia.
Ci riferiamo in particolare al software commerciale, dai prodotti del valore di poche migliaia di lire (come i giochi) fino a quelli dal costo molto più elevato dell'area gestionale e professionale.
È quindi evidente come questo problema riguardi un po' tutti, compresi i lettori di **Personal Software** che pure sono sempre stati incoraggiati a crearsi i propri programmi.
Abbiamo finora parlato del fenomeno, come di un problema e non ci sembra di esagerare: la circolazione di programmi copiati o comunque diffusi illegalmente è solamente la punta di un iceberg molto più vasto. Il vero nodo è costituito, a nostro giudizio, da uno errato approccio fatto da alcuni operatori del settore, soprattutto rivenditori.
In molti casi, ancora oggi, si considera in termini di sopportazione se non di fastidio vero e proprio, il fatto di dover rivendere del software. In pratica, lungi dal considerare tale attività fonte di possibili guadagni e quindi commercialmente remunerativa, la si intende semplicemente come un male necessario alla vendita dei computer.



In definitiva si rinuncia a priori ad investire e guadagnare sul software, così come lo si fa con l'hardware, e si ricorre a mezzi di vario tipo per fornire il cliente.
È chiaro che questo atteggiamento, anche se piuttosto diffuso, non costituisce di certo la norma: sono tantissimi, senz'altro la maggioranza, i rivenditori che operano correttamente e si rendono quindi conto del ruolo fondamentale che gioca il software.
D'altra parte, diffondere abusivamente del software, a parte l'ovvia illegalità dell'azione, è molto spesso svantaggioso. Infatti chi regala software copiato ha una perdita diretta in termini di tempo e supporti magnetici impiegati oltre alla perdita, molto più grave economicamente, derivante dalla mancata vendita.
Chi invece rivende a metà prezzo, o meno, dei programmi copiati (succede, succede ...), ha un guadagno tutto sommato paragonabile a quello che avrebbe vendendo i pacchetti originali, ma in cambio, una netta perdita di immagine agli occhi dello stesso cliente, danno non quantificabile, ma senza dubbio rilevante.
Molti obiettano che ragionamenti di questo tipo non tengono conto dell'acquirente, che in definitiva è colui che paga e che ovviamente è interessato a spendere il meno possibile. Questo è vero, ma è proprio nell'interesse dell'utente finale che occorre adottare politiche di rivendita rigorose.
Se, infatti, è senza dubbio vero che ottenere del software gratuitamente è nell'immediato la soluzione ideale, è altrettanto vero che in questo modo si crea uno sbilancio economico: quel pacchetto, in realtà, ha un costo derivante dal lavoro di chi lo ha prodotto e distribuito. Un mancato ritorno economico condizionerebbe ovviamente le produzioni future dei diversi autori.
Che si tratti di una situazione anomala lo dimostra anche il fatto che tutte le persone che vendono o acquistano software copiato, sicuramente si stupirebbero molto se un libraio offrisse loro la fotocopia di un libro piuttosto che una copia originale.
È ovvio che tutti questi discorsi sono diretti a chi imposta sul software copiato la propria attività commerciale: per l'utente finale che regala una copia del suo programma all'amico si può certamente chiudere un occhio, anche se l'operazione non è certo lecita. Ma, d'altra parte, scagli la prima pietra chi, in vita sua, non ha mai fotocopiato un libro! ■



Libri firmati JACKSON



Maurizio Piccoli
FENDER
storia di un mito (1945-1985)

Sono trascorsi quarant'anni da quando Leo Fender iniziò la sua magnifica impresa; questo libro, che pure non ha tratto spunto da ragioni celebrative, sicuramente è maturato dalla consapevolezza che un arco così ampio di tempo richiedesse un'analisi e un ordinato recupero della non indifferente mole di dati disponibili sulla globalità della produzione Fender.
Il libro abbraccia tutto ciò che dal 1945 ai giorni nostri è uscito con il marchio Fender, privilegiando adeguatamente quegli strumenti di maggior interesse sui quali si puntano gli occhi dei fans della casa americana.

249 pagine
codice **800H L. 28.000**

Goffredo Haus
ELEMENTI DI INFORMATICA MUSICALE

Questo libro è rivolto a chi intende accostarsi all'informatica musicale ed in particolare agli studenti universitari di Scienze dell'Informazione, Matematica, Fisica e Ingegneria Elettronica.
È rivolto anche al musicista interessato alle tematiche dell'informatica musicale, che disponga di un corredo elementare di nozioni sugli elaboratori elettronici.
Gli argomenti trattati comprendono: elementi di matematica di base; descrizione formale di fenomeni musicali; tecniche di analisi, elaborazione e sintesi del testo musicale; tecniche di analisi, elaborazione e sintesi del suono; metodi per la rappresentazione grafica di informazioni musicali; elementi di ingegneria del software musicale; elementi sulle architetture dei sistemi per l'elaborazione musicale; elementi sulle tecnologie avanzate utilizzate nel settore.

232 pagine
codice **802H L. 22.500**

La Biblioteca che fa testo

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 3.000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ____________________
n° ____________________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura
Partita I.V.A. | | | | | | | | | | | |

ORDINE MINIMO L. 50.000

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

GRUPPO EDITORIALE JACKSON

nuovidea

Ancora sul linguaggio macchina del Commodore 64

Sono un abbonato di **Personal Software**, e desiderando imparare a programmare in linguaggio macchina sul C 64, ho letto gli articoli in proposito di Alessandro Guida, oltre al Programmer Reference Guide della stessa Commodore. È tutto chiaro, tranne un piccolo particolare: che nessuno dice come si fa a programmare.

Dice Guida: batti Sys 828 e vai alla subroutine. E come si mette in macchina la subroutine? In BASIC si mettono frasi di istruzione, numerate progressivamente: che cosa corrisponde a ciò nel linguaggio macchina? Ho provato ad inserire i codici del programma riportato a pagina 59 della parte terza degli articoli di Guida, andando per tentativi, ma il computer non li capisce. E allora?

Mi sembra che l'argomento venga trattato come se si volesse insegnare a un selvaggio de Mato Grosso come si fa a guidare l'automobile, spiegandogli il motore dei minimi dettagli, ma senza dirgli che per guidare bisogna prima entrare nelle macchine e prendere in mano il volante. E del resto, questa è un'impressione che si ha spesso, leggendo la letteratura divulgativa (o che dovrebbe esserlo) del ramo.

Modestamente, ammetto di essere, nei confronti del computer, come quel selvaggio: ma se si scrive un articolo *Impariamo il linguaggio macchina* non bisogna supporre che il lettore sia proprio in queste condizioni? Perché chi lo sa già non ha bisogno di leggerlo, trattandosi di cose semplicissime una volta appurati certi punti chiave.

In conclusione, potrei avere quello che conta, più di mille parole complicate, e cioè il listao di un piccolo programma in linguaggio macchina, così come devo batterlo, punto per punto, sulla tastiera?

Giorgio Zaza
Rosignano Solvay (LI)

Ridurre in poche puntate su una rivista un argomento di questa portata, cercando di mantenere un linguaggio il più possibile semplice, è sicuramente un'impresa complicata. Nonostante tutto riteniamo che Guida ci sia in gran parte riuscito e d'altronde lei stesso afferma che è tutto chiaro.

Riguardo ai suoi dubbi relativi al metodo da utilizzare per inserire i programmi in linguaggio macchina, riteniamo che possano essere risolti da una attenta lettura delle due prime puntate pubblicate rispettivamente nei numeri 15 e 17 di **Personal Software**. *Nel numero 18, è stato pubblicato un programmino in BASIC che consente di introdurre i codici relativi a un programma in linguaggio macchina.*

Questa procedura è resa obbligatoria dal fatto che il C 64, in configurazione base, è programmabile soltanto in BASIC; per poter caricare ed eseguire programmi in linguaggio macchina occorre sfruttare le istruzioni BASIC, Peek, Poke e Sys.

POSTA

Niente paura

Leggo da molto la vostra rivista e posseggo un Commodore 64 più floppy disk e sono molto fiero di questo computer.
Mi preoccupa molto l'arrivo dei nuovi Commodore in Italia, chiedo perciò delle spiegazioni su tre motivi:
- il mio computer verrà ancora prodotto?
- se non venisse prodotto ci sarà ancora del software?
- nel caso la produzione cesserà mi conviene passare ad un sistema superiore?

Vorrei porvi anche un'altra domanda, a Natale mi regaleranno un monitor: quale marca mi consigliate con queste caratteristiche:
- a colori;
- con volume;
- non tanto ingombrante;
- che costi sulle 600.000 lire.

Vania Panizza
Milano

Come è noto, poco prima di Natale la Commodore ha iniziato la commercializzazione dei nuovi modelli C16 e Plus 4. Non ci risulta nessuna comunicazione riguardo ad una eventuale cessazione della produzione di C 64. Anzi, voci sicuramente attendibili, danno per certa una lunga vita al popolare Commodore.
D'altra parte, il successo che tuttora riscuote, non consentirebbe di certo a breve termine una sua uscita di produzione.
In ogni caso, quando presto o tardi il C 64 non verrà più prodotto, si verrà automaticamente a creare una situazione per cui l'altissimo numero di unità in circolazione, determineranno comunque una notevole richiesta di software. Questo fatto tenderà ad allungare la vita del C 64 in modo rilevante anche se non completamente prevedibile data l'assenza di altri casi della stessa portata.
Da quanto detto può sicuramente intuire il nostro consiglio relativamente al fatto di cambiare o meno calcolatore: lo sostituisca se i nuovi modelli o altre macchine di altre marche sono in grado di soddisfare esigenze cui il suo attuale computer non fa più fronte. Ma se, come ci sembra di capire dalla sua lettera, il cambio avverrebbe solo per stare al passo con le novità, ci sentiamo di sconsigliarla: il C 64 è tuttora un'ottima macchina in grado di dare grosse soddisfazioni ai suoi utilizzatori.
Anche se Natale è già trascorso le rispondiamo a proposito del monitor: riteniamo che la scelta più ovvia sia il modello 1701 della stessa Commodore che funziona molto bene e possiede tutte le caratteristiche indicate, compreso il prezzo.

■ ■ ■

Parliamo del Sega

Sono un vostro lettore e quindi voglio farvi tutti i miei complimenti per questa splendida rivista. Devo dirvi, però, che fra tutti i pregi che avete, c'è un difetto: perché non pubblicate programmi per Sega SC-3000, che è un ottimo computer?
Vi saluto e vi auguro una lunga vita.
P.S. - Posso mandarvi i miei programmi per Sega SC-3000?

Andrea De Luca
Torre Annunziata (NA)

La ringraziamo per i complimenti e prendiamo atto di quello che secondo lei è un difetto. In realtà si tratta di una necessità, in quanto non abbiamo ancora ricevuto programmi abbastanza validi da pubblicare.
Siamo d'accordo con lei che il Sega SC-3000 è un ottimo computer e quindi la invitiamo a mandarci i suoi programmi che provvederemo a pubblicare se di interesse generale. Buon lavoro!

■ ■ ■

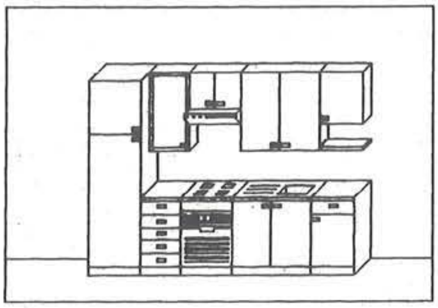
PERSONAL NEWS

Vitalità Sinclair

Da Londra e dall'Italia una serie di notizie riguardanti prodotti ed applicazioni per le macchine di Sir Clive. Una novità, ormai crediamo già disponibile nei negozi italiani, è lo ZX Spectrum+ 48 Kbyte, la versione potenziata e cresciuta del popolare home computer della Sinclair Research, presentato in Inghilterra alla fine del 1984. Ha una tastiera finalmente simile a quella delle macchine per scrivere, anche se i progettisti si sono presi un po' di libertà nel disporre alcuni tasti, e viene distribuito con un interessante corredo autodidattico. Assieme ad un manuale appositamente realizzato da una casa editrice inglese specializzata nella divulgazione dell'informatica, viene fornita una cassetta introduttiva di nuova concezione della stessa Dorling Kindersley. ZX Spectrum+ è pienamente compatibile con tutto il software prodotto per i predecessori della stessa serie e adotta anche le medesime periferiche. L'inclinazione della tastiera può essere modificata ed è stato previsto un tasto per resettare il computer senza necessità di spegnerlo. Il sistema di espansione consiste nello ZX Microdrive e nell'Interfaccia 1 che lo controlla e permette la connessione di altre periferiche.



Venendo a casa nostra, ecco Laser, un programma per sviluppare, mediante ZX Spectrum ,qualsiasi tipo di sistema integrale condizionato. Il collegamento alla stampante Totovelox permette poi di far stampare le schedine contestualmente allo sviluppo. Il funzionamento è guidato da menu e consente quindi un semplice uso anche a chi non ama molto le tastiere; è protetto contro errori accidentali e rifiuta le risposte errate o non previste. I criteri di base sono quelli adottati da tutti i sistemisti e consentono l'individuazione di "sezioni" di partite e la correzione di errori differenziata ed a base ibrida, nonchè l'applicazione di controlli statistici. Realizzato in Assembly per guadagnare in efficienza, il programma della Totocomputer raggruppa al termine, le colonne selezionate in sistemini integrali, permettendo così una certa economia del numero di schedine da stampare e convalidare.





Nel campo della progettazione ecco invece Grafex, Graphic Management System, un sistema grafico interattivo per il trattamento di archivi ad elementi modulari con contenuti informativi di tipo numerico, descrittivo e grafico. Realizzato dalla Capware di Ottaviano e distribuito in Italia dalla GBC, divisione Rebit, Grafex non necessita di elevate capacità di memoria e può quindi "girare" anche su microcomputer, trasformando magari anche uno Spectrum in un sofisticato e semplice strumento di lavoro. Le applicazioni? Architettura, impianti, image processing, medicina (analisi di elettrocardiogrammi, ...), arredamento, industria, hobby (modellismo ferroviario, trattamento di oggetti componibili).

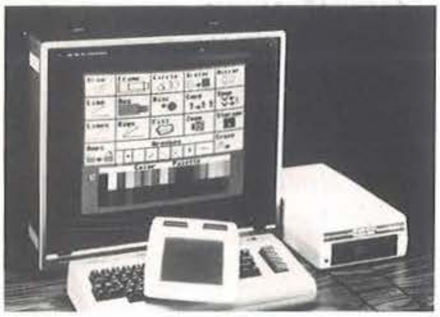
Rebit Computer
Viale Matteotti, 66
20092 Cinisello Balsamo (MI)
Tel. 02-6181801

Ulteriori informazioni:
Columbia Marketing
Tel. 02-77981

Tavoletta grafica per Apple, Commodore 64 e IBM PC

Per la distribuzione Telav sono disponibili in tutta Italia i prodotti Koala per la grafica e la didattica. Cominciamo in questo numero a proporre la tavoletta grafica e la penna ottica, rimandando ad una

successiva occasione la presentazione di altri interessanti prodotti. Definito da diffuse riviste estere, come uno dei migliori sistemi grafici disponibili, KoalaPad Touch Tablet è un semplice insieme di strumenti e programmi per trasformare un elaboratore casalingo in una stazione grafica di discrete prestazioni. La tavoletta è stata realizzata inizial-





mente per macchine basate sulla CPU 6502, quali Apple IIe e IIc, Atari e Commodore 64, ma oggi viene fornita anche per l'IBM PC. È un'appendice "magica" del computer: quando si disegna col dito o con una punta sulla tavoletta, la stessa immagine appare sullo schermo. E' possibile usare la stessa tavoletta per modificare i parametri e gli attributi del disegno o ottenere l'effetto "zoom" per rifinire i particolari. La versione Commodore è distribuita con tre programmi: Koala Painter, per il disegno sullo schermo, Koala Printer per tracciare disegni capaci di 16 sfumature di grigio con otto modelli diversi di stampanti e Instant Programmer's Guide per scrivere applicazioni basate su KoalaPad in BASIC o in Assembly. Per l'Apple II sono disponibili Koala Painter e Graphics Exhibitor. Quest'ultimo è un potente programma per realizzare immagini corredate di testo e utilizzare il personal di Cupertino come sistema dimostrativo audiovisivo, proponendo in continuo una serie di schermate precedentemente preparate. La penna ottica Gibson è un altro notevole prodotto per creare e manipolare oggetti direttamente sullo schermo. Corredata di quattro programmi Koala, permette di realizzare disegni animati o di creare musiche direttamente sul pentagramma che appare sullo schermo dell'Apple IIe. Per il Commodore 64 sono invece disponibili due programmi di corredo al sistema Gibson per disegnare con la penna ottica.

Telav International S.r.l.
Via Leonardo da Vinci, 43
20090 Trezzano S/N (MI)
Tel. 02-4455741

Tecnologia avanzata IBM

Allargando un po' il tiro, rispetto agli argomenti della nostra rivista, ricordiamo che l'IBM ha annunciato e comincia a distribuire il personal computer AT (Advanced Technology), basato su un microprocessore Intel 80286. Corredato della versione 3.0 del DOS, il PC AT può essere utilizzato contemporaneamente da un massimo di tre utenti e dispone per questo scopo di IBM PC Xenix, la versione Microsoft di Unix. La memoria di massa può raggiungere i 41 milioni di caratteri, mediante l'impiego di floppy capaci di 1.2 Mbyte e dischi fissi da 20 Mbyte. La memoria centrale può essere estese fino a 3 Mbyte, facilmente indirizzabili dai registri a 24 bit dell'80286 che adotta invece una parola di 16 bit per i dati. La velocità di esecuzione è almeno doppia (arriva anche a tre volte tanto) rispetto agli IBM PC della prima generazione. Il prezzo della configurazione base di questa macchina, dotata di chiave per evitare intrusioni non autorizzate, non supera i dieci milioni di lire.

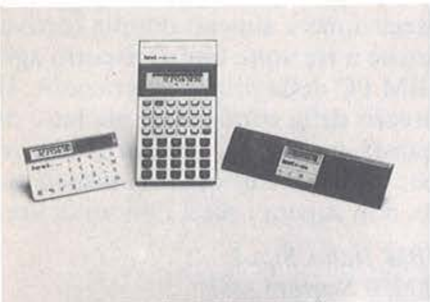
IBM Italia S.p.A.
20090 Segrate (MI)
Tel. 02-75484550

Pokerissimo di calcolatrici portatili

La Levi, che ha concluso in Dicembre il 75mo anno dalla sua fondazione, presenta cinque nuovi modelli di macchine da calcolo per le varie esigenze dell'ufficio e dello studio. Non tutte sono dotate di capacità di programmazione, ma hanno delle caratteristiche che le rendono interessanti anche per i nostri lettori. La VS 1202 è una calcolatrice da ufficio portatile, a 12 colonne con stampan-



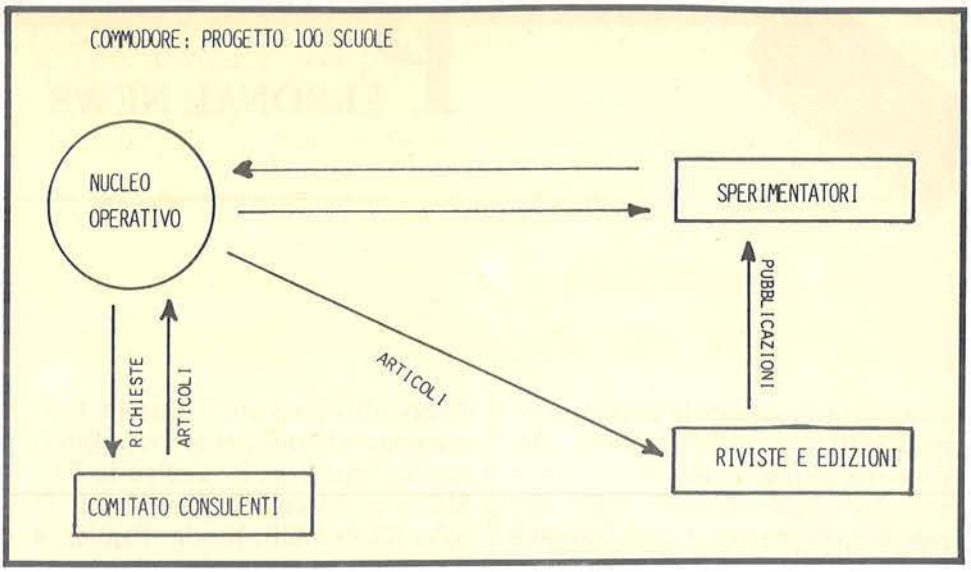
te incorporata. I calcoli IVA sono preprogrammati e prevedono tre arrotondamenti. La TP 1106 è molto sottile, ma conserva i grandi tasti tipici delle applicazioni professionali di questo genere di strumenti. Anch'essa alimentabile a batterie o a rete, ha una capacità di 10 cifre ed è dotata di una stampante termica veloce. La Levi F 106 a batterie solari, ha la forma di un righello e dispone di una scala graduata in centimetri e pollici: spessa solo 2,5 mm può essere un interessante completamento del corredo di lavoro della scrivania. Nel campo dell'extrapiatto c'è anche la F 103. Sempre a celle solari e sul modello delle carte di credito, non supera i due millimetri di spessore. Per gli amanti della programmazione e del debug, ecco l'ultimo acuto del gruppo, la PSR 198, a 10 cifre e capace di eseguire calcoli e conversioni tra sistemi di numerazione decimale, esadecimale ed ottale. Prevede anche trasformazioni tra diversi sistemi di misura.



E. Levi & C. S.p.A
Via Giambellino, 11
20146 Milano
Tel. 02-4220960

Commodore e la scuola

L'informatica fin dai primi anni di scuola. Come e perche? La Commodore Italiana cerca di contribuire al dibattito in corso su questo scottante tema, mediante il progetto LUCAS (L'Uso del Computer nellA Scuola dell'obbligo) che è ormai in atto da qualche mese in dieci scuole elementari o medie della provincia di Milano. Ogni istituto coinvolto nel progetto messo a punto dal COGI, Centro Orientamento Giovani, ha ricevuto dalla Commodore un laboratorio didattico per la sperimentazione. La stessa società ha istituito borse di studio per i docenti sperimentatori e gli esperti universitari che verificano e coordinano le loro attività. Al termine dell'anno scolastico una serie di relazioni illustrerà i risultati delle esperienze in queste scuole "campione", parlando dell'impatto con discipline quali l'educazione linguistica, tecnologica e scientifico-matematica. La stessa Commodore Italiana ha poi autonomamente avviato il progetto 100 scuole per mettere a disposizione di altrettante elementari e medie del nostro paese, laboratori di informatica costituiti da quattro calcolatori, memoria di massa, printer plotter, software specifico e supporto didattico a titolo di comodato gratuito. Ad ogni insegnante coinvolto nel progetto viene richiesto un ritorno sotto forma di relazioni periodiche sul lavoro svolto. La Commodore invita, comunque, tutti coloro che stanno utilizzando l'elaboratore per attività educative, a mettersi in contatto con il suo Ufficio Scuola per uno scambio di informazioni e una collaborazione che può essere molto proficua per entrambe le parti.





Commodore Italiana S.p.A.
Via Fratelli Gracchi, 48
20092 Cinisello Balsamo (MI)
Tel. 02-61832

Hit Parade Mastertronic

I giochi di produzione inglese hanno già ottenuto un buon successo nel nostro paese, tanto che nel solo mese di Settembre il distributore dichiara di averne venduti 15.000. Vi proponiamo l'elenco dei primi 10, secondo le preferenze degli acquirenti.

Videogioco	Home/personal computer
1: Duck Shoot	VIC 20
2: BMX Racers	Commodore 64
3: Space Walk	Commodore 64
4: 3 D Maze	VIC 20
5: Duck Shoot	Commodore 64
6: Mind Control	Commodore 64
7: Vegas Jackpot	VIC 20
8: Bullet	VIC 20
9: Rifle Range	Spectrum 16/48 Kbyte
10: Dark Star	Commodore 64

Persona
Tel. 045-592960

Carla Caccia
Tel. 02-5455813

Leoni informatica

HARDWARE & SOFTWARE HOUSE

HOTLINE

linea telefonica dedicata alla risoluzione dei problemi dei clienti. Chiamando il numero telefonico riservato che troverete sulla cartolina garanzia acclusa ai programmi, riceverete tutte le informazioni che vi necessitano.

UPDATE

servizio di aggiornamento continuo dei programmi acquistati. Ogni modifica ai programmi realizzati dalla Leoni Informatica sarà fornita agli utenti degli stessi.

GARANZIA

tutti i programmi Leoni Informatica sono coperti da garanzia a Vita contro guasti di origine.

COMMODORE 64 SOFTWARE

SOFTWARE PER COMMODORE 64 E PLUS/4

Programmi in configurazione base (*) IVA esclusa

Cod.	Descrizione	Prezzo	Cod.	Descrizione	Prezzo
		PERSONALI			
0046	Ammortamento Mutui	60.000	0049/T	Totoplus	100.000
0050/T	Totocalcio a sviluppo colonnare	80.000	0051/T	Gestione dei Conti di casa	100.000
0058/T	Calcolo dell'equo canone	80.000	0055/T	Impariamo il BASIC	100.000
0059/T	Modello 740	10.0000	0063	Cento programmi BASIC	80.000
0066	Conto corrente	100.000	0091/T	Rubrica-telefonica	100.000
0176	Diary 64 (Commodore)	95.000	0174/T	Corso di Dattilografia	80.000
		GESTIONI GENERALI			
0047	Anagrafiche	150.000	0056	Dichiarazione I.V.A.	60.000
0065/T	Fido Clienti	100.000	0067	Piano dei Conti (cli/for/gen)	100.000
0068	Appuntamenti	100.000	0071	Ordini (cli/for)	100.000
0090	Mailing List (riordino alfabetico cap. prov.)	100.000	0094	Scheda 4800 car. (cli/for/rapp./az. etc.)	100.000
0096	Scheda 4800 car. agganciata al Mailing	150.000	0097	Super Mail (5 chiavi accesso, riordini..)	180.000
0116	Scadenziario effetti (ric. bancarie, tratte, etc.)	100.000	0120	Contabilità fatture (Iva, impon. etc.)	100.000
0121	Contabilità Semplice (Tratte/Fatt./Conti/etc.)	250.000	0160	Bolle e Fatture	200.000
0125	Contabilità Generale (132 colonne)	300.000			
		GESTIONI SPECIFICHE			
0045	Agenti e Rappresentanti	150.000	0048	Scadenziario premi e polizze	150.000
0164	Agenzie Immobiliari	150.000	0086	Librerie e biblioteche	120.000
0148	Studi Ottici	200.000	0151	Farmacie	300.000
0149	Studi Dentistici	200.000	0152	Studi Medici	200.000
0131	Hotel e Pensioni	280.000	0132	Parrucchieri	280.000
0133	Gommisti	280.000	0134	Clubs Nautici	280.000
0135	Officine	280.000	0171	Ristoranti	280.000
0170	Tavola Calda	280.000	0172	Lavanderie	280.000
0175	Condominio New (132 colonne)	400.000			
		GESTIONE TESTI			
0190	Hes Writer	70.000	0191	Word Processor III	100.000
0192	Bank Street Writer	70.000	0319	Easy Script/T (Commodore)	75.000
		TECNICI			
0136	Legge 373 (calcolo degli isolamenti termici)	100.000	0140	Ingegneria civile I (calcoli strutt.)	100.000
0141	Ingegneria civile II (travi intelaiate)	100.000	0404	Computo metrico	200.000
0409	Diagnostica C64	40.000	0322	Doctor 64	60.000
		MAGAZZINI			
0142	Magazzino e Fatturazione semplici	100.000	0143	Magazzino Grossisti (2500 art.)	280.000
0144	Magazzino e Fatturazione agganciati	200.000	0158	Magazzino Dettaglio (2500 art.)	280.000
0148	Magazzino codice alfanumerico (600 art.)	200.000	0159	Magazzino Taglia/Col. (2500 art.)	280.000
		LINGUAGGI & UTILITIES			
0162/T	Screen Grafix (Abacus)	85.000	0163	Copia Disco singolo	50.000
0064	Petspeed Compiler (Commodore)	80.000	0165/T	Zoom	70.000
0167	Simon's Basic (Commodore)	85.000	0168/T	Turbo Tape	50.000
0177	Pilot (linguaggio)	70.000	0178	Ultra Basic	125.000
0179	Comp/Scompactor	50.000	0193/C	Basic 4.0	80.000
0194	Sprite Generator	70.000	0195	Assembler	80.000
0196	S.A.M. (Tronix)	100.000	0197	G-Pascal	95.000
0198	Forth 64 (Commodore)	95.000	0199	Tol 64	85.000
0200	Master (Commodore)	145.000	0201/C	Scheda CP/M (Commodore)	125.000
0333	Clone machine	100.000	0334	Unguard	200.000
0335	Fast Copy 4.5	100.000	0336	Music Composer	150.000
0337	Basic Programm Generator	250.000	0338	Copy Files	80.000
0320	The MANAGER 64	100.000	0321	Austro Compilatore	80.000
0210	Lite Pen (incluso sotfware)	130.000			
		GESTIONE DATI			
0157	Easy Calc	125.000	0205	Super Base	175.000
0206	Magic Desk (Commodore)	75.000	0207	Koala Joystick	100.000
0209	Data Log.	120.000	0220	Easy Calc Tape	125.000
0400	Stock Control	120.000	0401	Easy Label	120.000

N.B. (*) Per configurazione base si intende Commodore 64, Floppy 1541, Stampante Commodore a 80 colonne.
I codici barrati 'T' sono disponibili anche su cassetta, quelli barrati 'C' sono Cartridge.

Leoni informatica

Direzione Commerciale • via Valsolda, 21
20143 MILANO - tel. 02/8467378

Negozio vendite al pubblico: Milano - Corso di Porta Romana, 123

Language Tutor per Spectrum 48 Kbyte

Impariamo le lingue con il computer!?

di *Mario Manuzzi*

Dopo tanti giochi per il piccolo-grande Spectrum ecco ora un programma un poco più serio.
Oggigiorno conoscere una o più lingue straniere è divenuto non solo utile ma necessario; in tutti i settori della vita moderna il continuo interscambio tra il nostro paese ed il resto del mondo pone continuamente a contatto con realtà che non parlano la nostra lingua, purtroppo così poco diffusa. È quindi chiaro che chi possiede lo strumento per entrare facilmente all'interno di queste realtà sarà sempre favorito nel suo cammino, qualunque strada egli decida di percorrere.
Il programma vuole dimostrare che anche da un piccolo computer come lo Spectrum è possibile ricavare un valido aiuto nello studio delle lingue.
Il principio alla base di questo lavoro è quello del computer-amico, che insegna facendo divertire e stimolando l'attenzione: PAL, che in inglese significa appunto amico, compagno, ma che può essere anche inteso come acrostico di "Play And Learn" (gioca ed impara).
In particolare il programma vuole costituire un piccolo aiuto durante uno dei momenti più noiosi nell'apprendimento delle lingue: quello della ripetizione e memorizzazione dei vocaboli stranieri.

Caratteristiche del programma

Il programma occupa esattamente 11.752 byte di memoria, mentre le variabili circa 28.400. Rimane poco

Listato 1. *Il programma Language Tutor.*

```
   1 LOAD ""CODE
   3 LOAD "" DATA l$(): LOAD ""
DATA w(): LOAD "" DATA p$()
   5 BORDER 1: PAPER 1: INK 7: C
LS
  10 PRINT "LANGUAGE TUTOR"'"████
████████████████████████████████
██████████████████████████████"
  12 PRINT '"8/9-1983 © Mario Ma
nuzzi"''"n.XX-1984 © PERSONAL SO
FTWARE": PAUSE 200
  40 LET pp=VAL l$(22999 TO 2300
0): LET voc=VAL p$(pp,5 TO 7): L
ET mm=0
  50 CLS : PRINT "1  Dizionario
Francese-Italiano"''"2  Dizionar
io Italiano-Francese"''"3  Entra
ineur linguistique"''"4  Le pend
u"''"5  Inserzione nuovi vocabol
i"
  51 PRINT '"6  Modifiche dati"'
'"7  Memorizzazione dati"''"0  S
TOP "
  55 PAUSE 0: IF INKEY$="0" THEN
 STOP
  60 IF INKEY$<"1" OR INKEY$>"7"
 THEN GO TO 55
  65 GO TO VAL INKEY$*1000
1000 CLS : PRINT AT 5,2; INVERSE
 1;"DIZIONARIO FRANCESE-ITALIANO
": PAUSE 50: CLS
1100 INPUT "Scrivi il vocabolo f
rancese "' LINE a$: LET a=LEN a$
-1
1110 IF a$="" THEN GO TO 1100
1150 PRINT AT 5,0; INVERSE 1;a$
1200 FOR n=1 TO pp: IF a$>=p$(n)
 THEN LET b=VAL p$(n,5 TO 7): LE
T c=VAL p$(n+1,5 TO 7): NEXT n
1300 FOR n=b TO c: LET aa=INT w(
n)
1320 IF a$=l$(aa TO aa+a) THEN G
O TO 1800
1350 NEXT n
1400 PRINT AT 12,4; FLASH 1;"VOC
ABOLO NON MEMORIZZATO": GO TO 18
90
1800 LET bb=(w(n)-aa)*100-1+aa
1810 IF a$<>l$(aa TO bb) THEN PR
INT '' INVERSE 1;l$(aa TO bb)
1820 PRINT '' INK 0; PAPER 6;l$(
bb+1 TO w(n+1)-1)
1890 PRINT #1;"Vuoi cercare altr
i vocaboli ?"'"   (s/n/a)"
1892 PAUSE 0: IF INKEY$="s" OR I
NKEY$="S" THEN CLS : GO TO 1050
1894 IF INKEY$="a" OR INKEY$="A"
 THEN PRINT AT 2,15; INVERSE 1;"
n.";n
1896 IF INKEY$<>"n" AND INKEY$<>
"N" THEN GO TO 1892
1900 GO TO 50
2000 CLS : PRINT AT 5,2; INVERSE
 1;"DIZIONARIO ITALIANO-FRANCESE
": PAUSE 50: CLS
2100 INPUT "Scrivi la parola ita
liana"' LINE a$: LET a=LEN a$-1
2110 IF a$="" THEN GO TO 2100
2120 PRINT AT 5,0; INVERSE 1;a$
2200 FOR n=1 TO voc: LET aa=INT
w(n+1)-1: LET bb=(w(n)-INT w(n))
*100+INT w(n)
2250 IF l$(bb TO bb+a)=a$ THEN G
O TO 2800
2290 NEXT n
2400 PRINT AT 12,4; FLASH 1;"PAR
OLA NON MEMORIZZATA"
2500 PRINT #1;"Vuoi cercare altr
```

Language tutor per Spectrum 48 Kbyte

spazio libero (circa 14.00 byte), che dovrebbe essere lasciato per consentire il regolare funzionamento delle varie routine. Data la sua estensione può essere caricato solo su Spectrum 48 Kbyte nonostante sia possibile un adattamento in versione ridotta per Spectrum 16 Kbyte.
In sintesi, il programma consiste di 2 parti fondamentali: una prima preordinata alla creazione di un file dei vocaboli che si vogliono memorizzare, ed una seconda che utilizza i vocaboli stessi. È inserito anche un dizionario bilingue.
Possono essere utilizzati blocchi di circa 1.000 vocaboli (con rispettive traduzioni), ma non può esservi in memoria più di un blocco alla volta. Oggi vocabolo può avere una lunghezza media di circa 23 caratteri ma, venendo questi compattati nel vettore L$(), non vi è alcuna limitazione né spreco di memoria per vocaboli più lunghi o più corti.

Descrizione del programma

La struttura del programma è molto lineare e comprende 7 opzioni principali (figura 1).
0-65 - Introduzione. Vengono caricati da nastro i caratteri speciali (vocali accentate, ecc.) ed il file con i puntatori, viene ricavato il numero di vocaboli del file stesso e stampato il menu. Alla linea 65 si simula il comando On, non presente nel BASIC Sinclair.
1000-2850 - Dizionario. La ricerca viene effettuata in modo diverso e assai più rapido dal francese, data la struttura particolare del file (vedi dopo). È consentito anche inserire chiavi di ricerca troncate nel finale. Ad esempio se il vocabolo cercato è "il vetro, il bicchiere" = "le verre", sono permesse chiavi di ricerca come "il vetro, il bicchiere" oppure "il vetro", "il vetr", "il ve", ecc. Premendo il tasto A si può ottenere la

```
i vocaboli ?"'"  (s/n/a)"
2510 PAUSE 0: IF INKEY$="s" OR I
NKEY$="S" THEN CLS : GO TO 2100
2515 IF INKEY$="a" OR INKEY$="A"
 THEN PRINT AT 2,15; INVERSE 1;"
n.";n
2520 IF INKEY$<>"n" AND INKEY$<>
"N" THEN GO TO 2510
2600 GO TO 50
2800 LET aa=INT w(n+1)-1: IF a$<
>l$(bb TO aa) THEN PRINT '' INVE
RSE 1;l$(bb TO aa)
2850 LET aa=INT w(n): PRINT '' P
APER 6; INK 0;l$(aa TO bb-1): GO
 TO 2500
3000 CLS : PRINT AT 5,2; INVERSE
 1;"ENTRAINEUR LINGUISTIQUE": PA
USE 50: CLS
3050 PRINT "1  Scelta casuale"''
"2  Inizio da un punto partico-"
'"    lare"
3052 PAUSE 0: IF INKEY$="2" THEN
 LET b=2: GO TO 3200
3054 IF INKEY$<>"1" THEN GO TO 3
052
3100 LET b=1: LET a=INT (RND*voc
+1): CLS : GO TO 3500
3200 CLS : PRINT "Ho in memoria
";voc;" vocaboli"
3250 INPUT "Da che vocabolo vuoi
 iniziare ?"',a: CLS : GO TO 350
0
3500 IF a>voc THEN PRINT FLASH 1
;AT 10,0;"NON HO PIU' VOCABOLI I
N MEMORIA"; FLASH 0;AT 21,0;"Pre
mi un tasto per continuare": PAU
SE 0: CLS : GO TO 50
3510 LET aa=INT w(a): LET bb=(w(
a)-aa)*100+aa: LET cc=INT w(a+1)
-1: LET c$=l$(aa TO bb-1)
```

```
3550 PRINT ''"Trova la parola fr
ancese "'"che traduce:"'' INVERS
E 1;l$(bb TO cc)
3580 INPUT "(a/?/ENTER/ STOP )"'
 LINE a$
3600 IF a$=c$ THEN PRINT ''"ESAT
TO !"''"La traduzione corretta e
'proprio"'' PAPER 6; INK 0;c$: G
O TO 3900
3610 IF a$="" THEN CLS : GO TO 3
910
3620 IF a$=" STOP " THEN GO TO 5
0
3630 IF a$="?" THEN GO TO 3950
3640 IF a$="a" THEN CLS : PRINT
, INVERSE 1;"n.";a: GO TO 3550
3650 LET c=LEN c$: LET a=LEN a$
3658 IF c<>a-1 THEN GO TO 3678
3660 FOR n=1 TO a: LET b$=a$(1 T
O n-1)+a$(n+1 TO a)
3670 IF b$=c$ THEN CLS : PRINT I
NVERSE 1;a$; INVERSE 0''"E' quas
i esatto,riprova (1+)": GO TO 35
50
3675 NEXT n
3678 IF a<>c-1 THEN GO TO 3698
3680 FOR n=1 TO c: LET b$=c$(1 T
O n-1)+c$(n+1 TO c)
3690 IF b$=a$ THEN CLS : PRINT I
NVERSE 1;a$; INVERSE 0''"E' quas
i esatto,riprova (1-)": GO TO 35
50
3695 NEXT n
3698 IF a<>c THEN GO TO 3750
3700 FOR n=1 TO a: LET b$=a$: LE
T b$(n)=c$(n)
3710 IF b$=c$ THEN CLS : PRINT I
NVERSE 1;a$; INVERSE 0''"E' quas
i esatto,riprova (1<>)": GO TO 3
550
```

Seguito listato Language Tutor.

```
3715 NEXT n
3750 IF a$<>l$(aa TO bb-1) THEN
CLS : PRINT INVERSE 1;a$; INVERS
E 0''"E' sbagliato, riprova": GO
 TO 3550
3900 PRINT #1;"Premi un tasto pe
r continuare": PAUSE 0: CLS
3910 IF b=1 THEN GO TO 3100
3920 LET a=a+1: GO TO 3500
3950 PRINT ''"La soluzione e':"'
' PAPER 6; INK 0;l$(aa TO bb-1):
 GO TO 3900
4000 CLS : PRINT AT 5,2; INVERSE
 1;"LE PENDU": PAUSE 50: CLS
4100 RESTORE 4950: LET d=0: LET
x=240: GO SUB 4900
4110 PLOT 238,98: DRAW 4,0
4120 LET a=INT (RND*voc+1): LET
aa=INT w(a): LET bb=(w(a)-aa)*10
```

```
1   Dizionario Francese-Italiano
2   Dizionario Italiano-Francese
3   Entraîneur linguistique
4   Le pendu
5   Inserzione nuovi vocaboli
6   Modifiche dati
7   Memorizzazione dati
0   STOP
```

Figura 1. *Il primo menu con le 7 opzioni principali.*

```
CHR$ 146 (C) : ç
0 0 56 64 64 64 56 96

CHR$ 148 (E) : é
16 32 92 34 60 32 30 0

CHR$ 152 (I) : î
24 36 0 24 8 8 28 0

CHR$ 160 (Q) : è
4 2 57 68 120 64 60 0

CHR$ 161 (R) : ê
24 36 28 34 60 32 30 0

CHR$ 162 (S) : â
28 34 12 2 30 34 30 0

CHR$ 164 (U) : û
16 40 0 68 68 68 56 0
```

Figura 2. *Alcuni suggerimenti per la creazione di caratteri grafici speciali. Sono indicati il numero del carattere, il tasto assegnatogli, il carattere stesso e gli 8 valori da pokare nell'area Udg (65368/65535). Si può usare un programma del tipo: 10 For n=0 To 7: Read nn/ 20 Poke Usr "c"+n,nn / 30 Next n / 40 Data 0,0,56,64,64,64,56,96. Qui il carattere creato è "ç" e gli 8 valori che lo caratterizzano sono contenuti nella linea 40, mentre la linea 20 (dopo Usr e le virgolette) contiene il tasto da assegnare (in questo caso C).*

```
0-1+aa: LET cc=INT w(a+1)-1
4130 LET a$=l$(aa TO bb): LET b=
LEN a$: IF b>15 THEN GO TO 4120
4140 LET b$=" ": FOR n=2 TO b: L
ET b$=b$+" ": NEXT n
4150 FOR n=1 TO b: IF a$(n)<>" "
 AND a$(n)<>"-" AND a$(n)<>"'" A
ND a$(n)<>"!" THEN PRINT AT 20,n
*2;"_"
4155 NEXT n
4160 PRINT AT 0,0: FOR n=65 TO 7
7: PRINT INVERSE 1;CHR$ n;" ";:
NEXT n
4165 PRINT '': FOR n=78 TO 90: P
RINT INVERSE 1;CHR$ n;" ";: NEXT
 n
4200 FOR n=1 TO b: IF a$(n)="è"
OR a$(n)="é" OR a$(n)="ê" THEN L
ET a$(n)="E"
4215 IF a$(n)="û" THEN LET a$(n)
="U"
4220 IF a$(n)="à" OR a$(n)="â" T
HEN LET a$(n)="A"
4225 IF a$(n)="î" THEN LET a$(n)
="I"
4230 IF a$(n)="ç" THEN LET a$(n)
="C"
4235 IF a$(n)="-" THEN LET b$(n)
="-"
4240 IF a$(n)="!" THEN LET b$(n)
="!"
4245 IF a$(n)="'" THEN LET b$(n)
="'"
4250 IF CODE a$(n)>96 AND CODE a
$(n)<122 THEN LET a$(n)=CHR$ (CO
DE a$(n)-32)
4260 NEXT n
4290 FOR n=1 TO b: PRINT AT 19,n
*2;b$(n): NEXT n
4300 INPUT "Prova una lettera (?
/!) "; LINE c$: IF c$="" THEN GO
 TO 4300
4310 IF c$="!" THEN RANDOMIZE :
GO TO 4300
4320 IF c$="?" THEN GO TO 4785
4330 LET c$=c$(1): IF CODE c$>90
 THEN LET c$=CHR$ (CODE c$-32)
4350 IF CODE c$<78 THEN PRINT AT
 1,(CODE c$-65)*2;"█"
4360 IF CODE c$>77 THEN PRINT AT
 3,(CODE c$-78)*2;"█"
4400 LET c=0: FOR n=1 TO b
4410 IF a$(n)=c$ THEN LET b$(n)=
c$: LET c=1
4420 NEXT n
4430 FOR n=1 TO b: PRINT AT 19,n
*2;b$(n): NEXT n
4450 IF a$=b$ THEN GO TO 4600
4460 IF c=1 THEN GO TO 4300
4470 IF d=8 THEN GO TO 4700
4500 LET d=d+1: READ x0,y0,x,y
4520 PLOT x0,y0: DRAW x,y: GO TO
 4300
4600 FOR n=0 TO 30: BEEP .01,n:
BEEP .01,30-n: NEXT n
4605 OVER 1: LET x=240: GO SUB 4
900
4610 PLOT 238,98: DRAW 4,0
4620 OVER 0: LET x=146: GO SUB 4
900
4630 PLOT 143,99: DRAW 6,0,PI/2:
 GO TO 4800
4700 FOR n=20 TO -10 STEP -1: BE
EP .01,n: NEXT n
4705 OVER 1: PLOT 255,35: DRAW -
48,0
4720 DRAW 0,-30: PLOT 238,98: DR
AW 4,0
4730 PLOT 255,87: DRAW -15,-15:
DRAW -15,15
4740 OVER 0: PLOT 236,51: DRAW 4
,21: DRAW 4,-21
4750 OVER 1: PLOT 255,36: DRAW -
15,15: DRAW -15,-15
```

Seguito listato Language Tutor.

```
4760 OVER 0: PLOT 236,30: DRAW 4
,21: DRAW 4,-21
4770 PLOT 237,97: DRAW 6,0,-PI/2
4780 PRINT AT 17,0;"La soluzione
 era :"
4785 FOR n=1 TO b: PRINT AT 19,n
*2; INK 0; PAPER 6;a$(n);" ": NE
XT n
4800 PRINT AT 20,0;"
                ";AT 21,0;"Vuoi g
iocare ancora ? (s/n)"
4805 PRINT AT 13,1;l$(bb+1 TO cc
)
4810 PAUSE 0: IF INKEY$="s" OR I
NKEY$="S" THEN CLS : GO TO 4080
4820 IF INKEY$<>"n" AND INKEY$<>
"N" THEN GO TO 4810
4830 GO TO 50
4900 CIRCLE x,102,8
4910 PLOT x+4,104: PLOT x-4,104:
 PLOT x,101
4920 PLOT x,93: DRAW 0,-20
4925 PLOT x,71: DRAW 0,-19
4930 PLOT x-15,36: DRAW 15,15: D
RAW 15,-15
4940 PLOT x-15,87: DRAW 15,-15:
DRAW 15,15: RETURN
4950 DATA 120,35,135,0,184,35,0,
91
4960 DATA 168,35,16,16,184,110,1
6,16
4970 DATA 184,126,68,0,184,110,1
6,16
4980 DATA 204,126,-20,-20,240,12
6,0,-16
5000 CLS : PRINT AT 5,2; INVERSE
 1;"INSERZIONE NUOVI VOCABOLI":
PAUSE 50: CLS
5100 INPUT "Scrivi il vocabolo f
rancese "' LINE a$
5110 PRINT ' INVERSE 1;a$
5120 INPUT "Scrivi la traduzione
 italiana"' LINE b$
5130 PRINT '' PAPER 6; INK 0;b$
5150 LET a=LEN a$: LET c=LEN b$+
a
5180 PRINT #1;"E' corretto ? (s/
n)"
5181 PAUSE 0: IF INKEY$="n" OR I
NKEY$="N" THEN CLS : GO TO 5050
5182 IF INKEY$<>"s" AND INKEY$<>
"S" THEN GO TO 5181
5190 INPUT "": PRINT AT 13,7; IN
VERSE 1;"ATTENDI UN ATTIMO"
5400 FOR n=pp TO 1 STEP -1: IF p
$(n)>a$ THEN LET cc=VAL p$(n,5 T
O 7)+1: LET p$(n,5 TO 7)=STR$ cc
: NEXT n
5500 FOR m=voc TO VAL p$(n,5 TO
7) STEP -1
5510 LET aa=INT w(m): LET bb=(w(
m)-aa)*100+aa-1
5520 IF a$<=l$(aa TO bb) THEN NE
XT m
5550 LET aa=INT w(m+1): LET b=IN
T ((w(voc+1)+1)/1000): LET d=INT
 (aa/1000+1)
5560 FOR n=b TO d STEP -1: LET l
$(n*1000+c TO (n+1)*1000+c)=l$(n
*1000 TO (n+1)*1000): NEXT n
5570 LET l$(aa+c TO (n+1)*1000+c
)=l$(aa TO (n+1)*1000)
5580 LET l$(aa TO aa+c-1)=a$+b$
5590 LET w(voc+2)=INT w(voc+1)+c
5600 FOR n=voc TO m+1 STEP -1: L
ET w(n+1)=w(n)+c: NEXT n
5650 LET voc=VAL p$(pp,5 TO 7):
LET w(n+1)=INT w(n+1)+a/100
5700 IF mm=1 THEN GO TO 6500
5800 PRINT AT 13,7; FLASH 1;"PAR
OLA MEMORIZZATA"
5810 PRINT #1;"Vuoi inserire alt
ri vocaboli ?"'"   (s/n)"
5820 PAUSE 0: IF INKEY$="s" OR I
NKEY$="S" THEN CLS : GO TO 5050
5840 IF INKEY$<>"n" AND INKEY$<>
"N" THEN GO TO 5820
5900 GO TO 50
6000 CLS : PRINT AT 5,2; INVERSE
 1;"MODIFICHE DATI": PAUSE 50: C
```

posizione del vocabolo nel file (cioè 1° vocabolo, 2° vocabolo, ecc.).
3000-3950 - Ripetitore. Chiede di trovare il vocabolo francese partendo da quello corrispondente italiano. È in grado di rilevare se vi è solo un piccolo errore (1 lettera in più, in meno o diversa). "?" dà la soluzione, "A" il numero di archivio, Stop ritorna al 1° menu ed Enter consente di passare al prossimo vocabolo, nel caso già si conosca la risposta e non si voglia perdere tempo a scriverla. È possibile scegliere a caso i vocaboli oppure partire da un punto preciso. Questa seconda possibilità permette di fermarsi quando si vuole e, dopo aver fatto un appunto del numero del vocabolo a cui si è giunti, riprendere in un secondo tempo da quella stessa posizione senza dover ricominciare tutto daccapo.
4000-4980 - L'impiccato. È una versione leggermente modificata e migliorata del gioco presentato in fondo al manuale di istruzioni, per cui non necessita di molte spiegazioni. Viene estratto a caso un vocabolo dall'archivio, modificandolo secondo necessità, qualora vi siano caratteri speciali o segni di interpunzione. "?" per conoscere la soluzione e "!" per "rimescolare le carte", per influire cioè sulla funzione di estrazione casuale.
5000-5900 - Inserzione. Può risultare lenta, poiché comporta il riordinamento dell'intero file. Per migliorare la situazione occorre agire sui puntatori in P$(). Notare che le linee 5550-5570 lavorano non sulla matrice intera ma su blocchi di 1.000 byte, poiché nell'area di lavoro non c'è spazio sufficiente per lavorare sull'intera matrice.
6100-6590 - Modifiche. Viene cancellato il vocabolo scelto ed eventualmente sostituito con uno nuovo, sfruttando la routine di inserzione, che viene segnalata dal flag MM.
6600-6799 - Modifiche. È una routine molto importante, che consente di abbreviare i tempi di ricerca e di inserzione. (vedi dopo). Enter al primo Input conferma: i vecchi puntatori, ad uno successivo ne ferma il numero (max. 30). Dopo la modifica delle stringhe di riferimento vengono calcolati i nuovi valori. Vicino ai puntatori attualmente in uso vengono indicate le stringhe di riferimento consigliate dal computer per quel numero di puntatori.
6800-6950 - Modifiche. Controlla se per distrazione non siano stati inseriti 2 vocaboli uguali, automaticamente o visualizzando l'intero file sullo schermo.
7000-7600 - Memorizzazione. Effetto automaticamente il Save ed il Verify dei dati.
9000-9020 - Inizializzazione del file. Prepara le variabili per la creazione di un nuovo blocco di dati.

Iniziare il programma

Una volta battuto con tanta pazienza il listato, lo si può registrare

con un Save "language FR" Line 0. Una volta ricaricato da nastro partirà automaticamente per caricare il file di vocaboli.
Per poter creare un file di vocaboli occorre però dare il Break ed iniziare con un Run 9000 (operazione necessaria all'inizio, dopo la battitura del programma, in quanto non esiste ancora il file). La stessa operazione è necessaria ogni qualvolta si voglia cancellare un vecchio blocco di dati e crearne uno totalmente nuovo.
Attenzione. Non dare mai comandi di Run o Clear, che cancellerebbero tutti i vocaboli in memoria. In caso di Break occorre ripartire con un Go To 50, che riporta al menu principale.
Una volta creato un file di vocaboli, o modificato un file preesistente, occorre salvarlo tramite l'opzione 7 del programma. È consigliabile usare una cassetta corta e registrare il programma ed i caratteri speciali da un lato, riservando l'altro lato per il file ed i puntatori, così da evitare il rischio di cancellazioni.

Il file e le variabili

I vocaboli sono strutturati in maniera un po' curiosa, per consentirne una estrazione più rapida ed una gestione più comoda. È un metodo che ha comunque vantaggi e svantaggi.
L$(): contiene tutti i vocaboli senza interruzione, in modo tale da permettere l'inserzione di vocaboli di qualunque lunghezza senza sprechi.
W(): contiene la posizione dei vocaboli in L$(). La parte intera del numero punta al primo carattere del vocabolo, la parte decimale indica la lunghezza del vocabolo francese. La lunghezza della traduzione italiana si ottiene facendo riferimento al primo byte del vocabolo successivo. Ad esempio se w(1)=1.11 e w(2)=26.10 il primo record inizierà dal 1° byte e finirà al 25° byte; il 1° campo sarà lungo 11 byte ed il 2° invece 14 (26 - (11+1)), andando dal 12° al 25° byte compresi.
P$(): contiene i puntatori. I primi 4 caratteri costituiscono le stringhe di riferimento ed i successivi 3 il numero di archivio del primo vocabolo successivo (in ordine alfabetico) alla stringa di riferimento.
Quando viene effettuata la ricerca dal francese, essendo il file ordinato alfabeticamente, non è necessario confrontare tutti i vocaboli con la stringa di ricerca, ma solo quelli compresi tra i 2 puntatori immediatamente superiori ed inferiori alla stringa stessa. Se cioè ad esempio la stringa di ricerca è "aller" e lo stato dei puntatori è P$(1)="aaaa001", P$(2)="1111030", P$(3)="zzzz072", è chiaro come i

Seguito listato Language Tutor.

```
LS
6050 PRINT "1  Cancella vocabolo
"''"2  Modifica vocabolo"''"3  M
odifica puntatori"''"4  Controll
a vocaboli"
6060 PAUSE 0: LET a=VAL INKEY$:
CLS : IF a=3 THEN GO TO 6600
6070 IF a=4 THEN GO TO 6800
6100 IF a=1 THEN PRINT INVERSE 1
;"CANCELLAZIONE VOCABOLO"
6110 IF a=2 THEN PRINT INVERSE 1
;"MODIFICA VOCABOLO"
6150 PRINT ''"Inserisci il numer
o di archivio del vocabolo."'"Se
 non lo conosci premi ENTER  e
cerca la parola nel diziona-  ri
o. Premi ""a"" per avere il nu-"
'"mero."
6160 INPUT , LINE a$: IF a$="" T
HEN GO TO 50
6200 LET b=VAL a$: LET aa=INT w(
b): LET bb=(w(b)-aa)*100+aa: LET
 cc=INT w(b+1)-1: LET c=cc-aa+1
6205 IF b>voc THEN GO TO 6160
6210 CLS : PRINT 'l$(aa TO bb-1)
''l$(bb TO cc)
6220 PRINT #1;"E' questo il voca
bolo ? (s/n)"
6225 PAUSE 0: IF INKEY$="n" OR I
NKEY$="N" THEN CLS : GO TO 6100
6230 IF INKEY$<>"s" AND INKEY$<>
"S" THEN GO TO 6225
6240 INPUT "": PRINT INVERSE 1;A
T 13,6;"ATTENDI UN ATTIMO"
6250 FOR n=1 TO pp: IF p$(n)>=l$
(aa TO bb) THEN LET d=VAL p$(n,5
 TO 7)-1: LET p$(n,5 TO 7)=STR$
d
6255 NEXT n
6260 LET d=INT (aa/1000+1): LET
e=INT (w(voc+1)/1000+1)
6270 LET l$(aa TO d*1000)=l$(aa+
c TO e*1000+c)
6280 FOR n=d TO e: LET l$(n*1000
 TO (n+1)*1000)=l$(n*1000+c TO (
n+1)*1000+c): NEXT n
6290 FOR n=b TO voc: LET w(n)=w(
n+1)-c: NEXT n: LET w(voc+1)=0
6300 LET voc=VAL p$(pp,5 TO 7)
6350 IF a=1 THEN PRINT AT 13,5;
FLASH 1;"VOCABOLO CANCELLATO": G
O TO 6900
6400 PRINT AT 13,6;"
     "
6450 PRINT AT 5,0: LET mm=1: GO
TO 5050
6500 LET mm=0: PRINT AT 13,5; FL
ASH 1;"VOCABOLO MODIFICATO"
6590 GO TO 6900
6600 PRINT INVERSE 1;"MODIFICA P
UNTATORI"
6610 PRINT '"Vocaboli ";voc,"Pun
tatori ";pp'"voc/pp ";voc/(pp-1)
''
6620 FOR n=1 TO pp: LET b=voc/(p
p-1)*(n-1): IF b<1 THEN LET b=1
6630 LET aa=INT w(b)
6640 PRINT p$(n,1 TO 4);"/";l$(a
a TO aa+3);" ";p$(n,5 TO 7),: NE
XT n
6650 FOR n=1 TO 30: INPUT "ENTER
 o nuovo p$(";(n);") "; LINE p$(
n)
6660 IF p$(1)="           " THEN LET
 p$(1)="aaaa1": PRINT FLASH 1;AT
 18,4;"PUNTATORI CONFERMATI": GO
```

Language tutor per Spectrum 48 Kbyte

VARIABILI IMPIEGATE

A$,B$,C$: stringhe di lavoro multiuso.
A,B,C,D,E: variabili di lavoro multiuso.
AA,BB,CC: variabili di lavoro. In genere indicano la posizione in L$() dei vari vocaboli.
N,M: variabili in genere utilizzate in cicli Next/For.
MM: flag che indica l'uso della routine di inserzione durante la modifica di un vocabolo.
PP: numero dei puntatori. È ripetuto in L$(22999 TO 23000).
VOC: numero dei vocaboli. È ripetuto in P$(PP,5 TO 7).
X,Y,X0,Y0: variabili usate per le posizioni di Plot e Draw.

due puntatori immediatamente superiori ed inferiori alla stringa di ricerca "aller" saranno il primo ed il secondo (aaaa<aller>1111). Di conseguenza verranno confrontati solo i vocaboli dal 1° al 30°, consentendo di ridurre notevolmente il lavoro necessario.

Fantasia e voglia di fare

Questo programma è stato preparato specificatamente per lo studio del francese, ma è comunque assai facile modificare il listato per renderlo adatto all'inglese, al tedesco, allo spagnolo od anche al russo, al greco od al latino. È possibile utilizzare il programma anche per lingue che fanno uso di caratteri diversi da quelli normali (ad esempio cirillici, greci), agendo sulla variabile di sistema Chars (23606), che consente di sostituire al normale set disponibile in ROM un nuovo set appositamente creato (eventualmente mediante un programma del tipo "Graphic Creator" della Llamasoft, studiato appunto per questo scopo). Lingue come l'inglese non necessitano invece nemmeno di altri caratteri speciali da creare in aggiunta a quelli normali (vocali accentate, ecc.), per cui è possibile cancellare senza timore, e con risparmio di memoria, ogni linea di programma destinata a gestire questi ultimi (ad esempio le linee 4215-4230).
Essendo poi il programma strutturato in blocchi pressoché autonomi è possibile eliminarne alcuni per sostituirli con nuovi. Forse il modo migliore di ovviare ai limiti di memoria è quello di dividere il programma in 2 sottoprogrammi separati: il primo preordinato alla costruzione del file ed il secondo alla sua utilizzazione.

Lo stesso file è poi senza dubbio riutilizzabile in nuovi programmi. Insomma la morale è sempre la stessa: fantasia e voglia di fare! Quanto visto vuol essere solo uno spunto per idee nuove. ■

Seguito listato Language Tutor.

```
 TO 6900
6670 IF p$(n)="       " THEN LET
pp=n: LET p$(1,1)=" ": LET p$(n)
="ûûûû"+STR$ voc: LET l$(22999 T
O 23000)=STR$ pp: LET n=30
6680 NEXT n
6700 FOR n=2 TO pp-1: FOR m=1 TO
 voc: LET aa=INT w(m)
6710 IF m=voc OR p$(n,1 TO 4)<=l
$(aa TO aa+3) THEN LET p$(n,5 TO
 7)=STR$ m: LET m=voc
6720 NEXT m: NEXT n
6750 CLS : GO TO 6600
6800 PRINT INVERSE 1;"CONTROLLO
VOCABOLI"
6810 PRINT '"1  Automatico"''"2
 Controllato": PAUSE 0: LET c=VA
L INKEY$: CLS
6830 FOR n=1 TO voc-1: LET aa=IN
T w(n): LET bb=(w(n)-aa)*100+aa-
1: LET a=INT w(n+1): LET b=(w(n+
1)-a)*100+a-1
6840 IF c=2 THEN PRINT AT 1,0;l$
(aa TO bb);"  "''l$(a TO b);"  "
: PAUSE 0
6850 IF l$(aa TO bb)=l$(a TO b)
THEN PRINT ''n;"  ";l$(aa TO bb)
''n+1;"  ";l$(a TO b)
6855 PRINT AT 0,28; INVERSE 1;n
6860 NEXT n
6890 PRINT FLASH 1;AT 13,5;"DATI
 CONTROLLATI"
6900 INPUT "": PRINT #1;"Vuoi mo
dificare altri dati ? "'"   (s/n)
"
6910 PAUSE 0: IF INKEY$="s" OR I
NKEY$="S" THEN CLS : GO TO 6050
6920 IF INKEY$<>"n" AND INKEY$<>
"N" THEN GO TO 6910
6950 GO TO 50
7000 CLS : PRINT AT 5,2; INVERSE
 1;"MEMORIZZAZIONE DATI": PAUSE
50
7100 SAVE "FR" DATA l$(): SAVE "
FR" DATA w(): SAVE "FR" DATA p$(
)
7200 CLS : PRINT AT 5,2;"RIPORTA
 INDIETRO IL NASTRO"'"  PER IL
CONTROLLO"
7220 PRINT '"  INSERISCI IL CAVE
TTO"''"  PREMI UN TASTO"
7250 PRINT #1;" Oppure N per pro
seguire": PAUSE 0: IF INKEY$="n"
 OR INKEY$="N" THEN GO TO 50
7350 CLS : PRINT AT 11,4; FLASH
1;"CONTROLLO REGISTRAZIONE"''
7500 VERIFY "" DATA l$(): VERIFY
 "" DATA w(): VERIFY "" DATA p$(
)
7600 CLS : PRINT AT 11,2;"REGIST
RAZIONE EFFETTUATA"'"         CO
RRETTAMENTE": PAUSE 100: GO TO 5
0
9000 DIM w(999): DIM p$(30,7): D
IM l$(23000): LET l$(22999 TO )=
"05"
9010 LET p$(1)="aaaa1": LET p$(2
)="eaaa1": LET p$(3)="maaa2": LE
T p$(4)="saaa2": LET p$(5)="ûûûû
2"
9020 LET l$(1 TO 46)="l'esclavag
ela schiavitu'le romarinil rosma
rino": LET w(1)=1.11: LET w(2)=2
5.1: LET w(3)=47
9100 GO TO 40
```

Sprite Editor

Utilizziamo la grafica evoluta per il nostro C 64

di *Chiara Tovena*

Introduzione

Tutti i sessantaquattrini sanno cosa sono gli sprite; quasi tutti i videogame e i giochi in generale ne fanno uso e certamente avrete già pensato di scriverne uno da voi.
Creare uno sprite però sembra essere un'impresa, stando almeno alle descrizioni che forniscono i manuali: matrici quadrettate, byte esadecimali, potenze di due, ecc...
Benvenuti quindi ai numerosi programmi utility che permettono di disegnarsi comodamente i propri sprite senza impazzire, con facilità di modifiche qualora il risultato non soddisfacesse. Anche il programma che vedremo di seguito è uno Sprite Editor, che ha il vantaggio di essere il più completo possibile di funzioni per permettere all'utente un lavoro veloce, e inoltre di essere molto rifinito.
Per caricare il programma, prima del Load, eseguite un:
Poke 44,16: Poke 4096,0
che serve a spostare più avanti i puntatori del testo BASIC, in modo da lasciare spazio, prima del programma stesso, per le sprite. Se vi dimenticate questi Poke il programma partendo se ne accorge e vi chiederà di ricaricarlo al posto giusto.
Se vi interessa solo usarlo, fatelo senza problema, ma se siete anche interessati alla buona programmazione (quanti programmi digitati con le dita ... dei piedi si vedono in giro!!) allora studiatelo anche, c'è una sezione apposta che lo descrive adeguatamente. Se qualche subroutine poi vi piace, o vi sembra particolarmente utile, inseritela nel vostro "archivio" personale di idee e usatela nei vostri programmi.

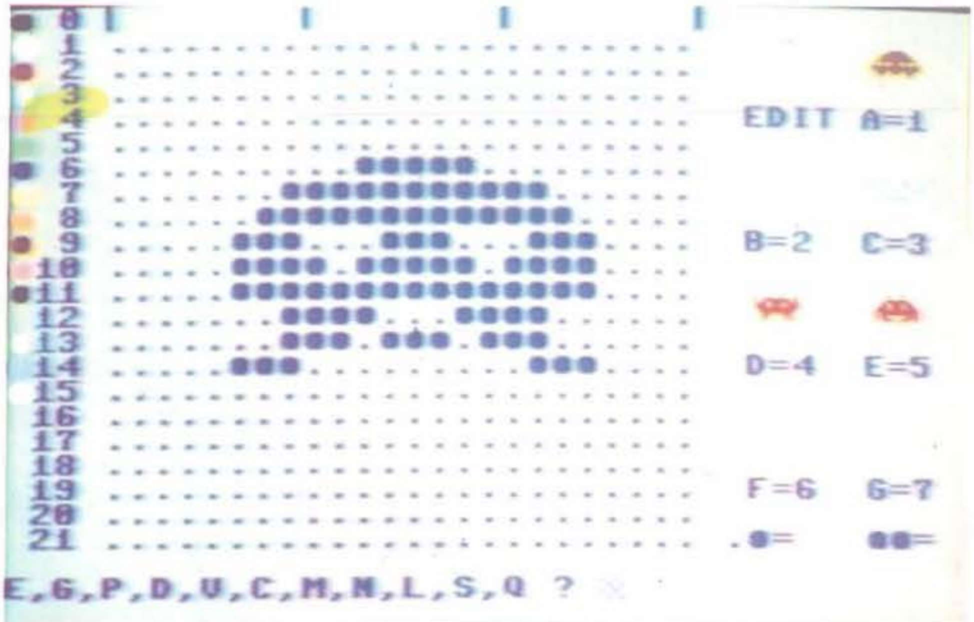


Figura 1. *Un esempio di figura.*

Cosa fa il programma e come si usa

Questo programma è stato creato per poter comodamente disegnare delle figure di sprite, poterle correggere, modificare vedendone subito il risultato, anche a colori, specie se si va in modo multicolor. L'uso è semplicissimo, tutte le operazioni sono pilotate da programma, in fase di scelta è sempre presente il menu, e si è cercato di sfruttare ogni dettaglio per rendere comoda la vita dell'utente (ad esempio i colori sono tutti visibili e numerati, ed altri particolari che indicano la cura con cui un vero programma di utility dovrebbe essere messo a punto).
Sul video compare anzitutto un reticolo di 24x21 puntini, con le tacche che delimitano i tre byte in orizzontale e una numerazione verticale che facilita il centraggio di un disegno. I pallini colorati che compaiono sulla sinistra servono a ricordare i colori disponibili e sono abbinati al loro numero di codice: l'utente non deve fare nemmeno lo sforzo di consultare il manuale!
Sulla destra si accendono gli otto sprite: facciamo subito una precisazione: si possono avere visibili contemporaneamente solo otto dei 32 sprite che invece è possibile tenere in memoria. Per non creare confusione noi chiameremo sprite i blocchetti di 64 byte posti in memoria, e figura i disegni visualizzati: così sulla destra si accendono 8 figure: la prima è quella di edit (cioè quella sulla quale state lavorando sul reticolo e corrisponde allo sprite n. 0) e le altre visualizzano gli sprite n. 1-7, ma vedremo che è possibile scegliere quale sprite vedere. Le figure sono tutte vuote, quindi del colore di fondo, e il

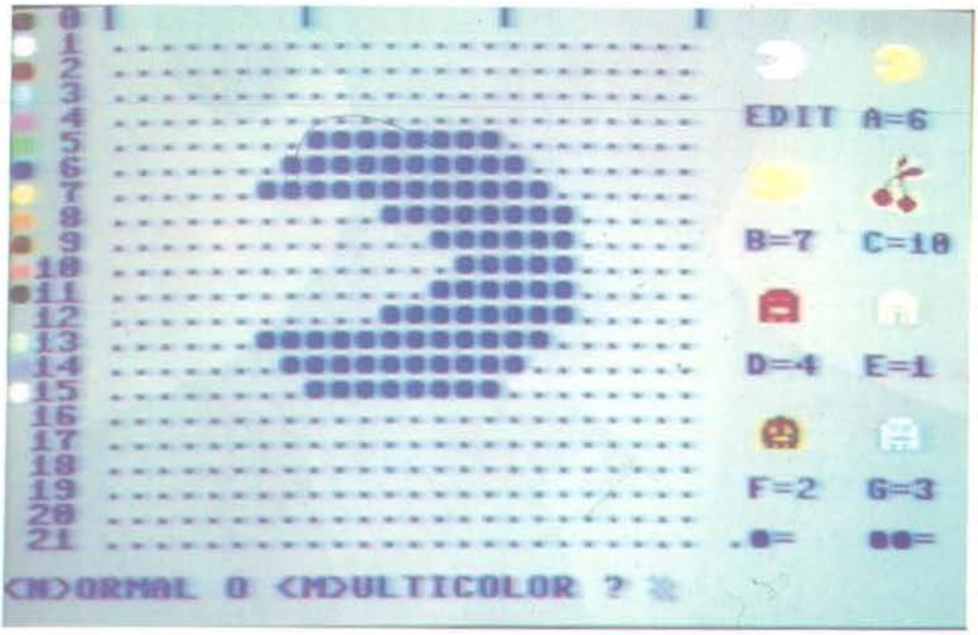


Figura 2. *Un altro sprite.*

colore che apparirà disegnandole è il bianco (con possibilità di essere cambiato più tardi).

Sotto ci sono i disegni che servono a ricordare i colori selezionati per il multicolor: il registro 0 (un puntino e un pallino adiacenti sul video), e il registro 1 (due pallini adiacenti).

In basso il menu: E,G,P,D, V,C,M,N,L,S,Q? e si può scegliere.

E (Edit): permette di introdurre un nuovo sprite: sul reticolo un puntino lampeggia e se premete <spazio> esso diventa un pallino e la figura di edit si accende nel punto corrispondente. Un altro <spazio> nella stessa posizione e il pallino si rispegne (comando toggle). Per spostarvi avanti e indietro premete i due tasti cursore avanti e cursore in basso, e per andare in alto e in basso premete A e Z. Se arrivate dal bordo della riga o della colonna ritornerete automaticamente all'inizio dall'altra parte. Man mano che costruite il disegno sul reticolo potete vederlo visualizzato sulla figura di edit, così da avere un riscontro immediato del risultato. Per uscire dallo stato di edit premere Q.

P (Put): trasferisce i byte degli sprite di edit in un altro dei possibili 31 blocchetti di dati. Questo perché lo sprite di edit è usato solamente per disegnare e non è possibile colorarla o salvarla su nastro. Bisogna allora spostarla in un altro posto: il programma chiede "Put in Sprite (1-31)? cioé dove devo metterlo? e si risponde col numero dello sprite voluto; i precedenti dati di quella sprite vengono naturalmente perduti. Battendo solamente un <Cr> si esce dalla routine senza toccare niente (se qualcuno ha premuto P per errore). La sprite comunque rimane anche sul reticolo di edit.

N (New): per ripulire il reticolo di edit dopo aver spostato uno sprite già disegnato. Il programma chiede e si desidera un reticolo tutto vuoto o tutto pieno: <E>mpty or <F>ill? In un caso cancellerà tutto, nel secondo accenderà tutti i punti dello sprite. Dipende se l'utento deve disegnare una figura per la maggior parte piena o vuota.

G (Get): per spostare uno sprite già disegnato nello sprite di edit e poterlo modificare ulteriormente. Il programma chiede: Get Sprite (1-31)? Cioè quale sprite volete editare? Come al solito battendo <Cr> si torna al menu senza fare niente. Lo sprite letto rimane anche nel suo posto, cioè non viene cancellato dalla sua posizione.

D (Data): ovvero come sapere i byte che compongono uno sprite. Dopo aver chiesto che sprite si desidera (Data of Sprite (1-31)?) il programma stampa sul reticolo, in coincidenza con le righe e le colonne appropriate, i 63 byte del blocchetto scelto. Lo sprite non compare automaticamente anche nella figura di edit. Quando avete finito di leggerli, o copiarli, battete <Cr> e i numeri scompaiono, facendo ricomparire il sottostante disegno di edit.

V (View): per poter visualizzare qualunque blocchetto dati.

Il comando serve a far puntare una certa figura su uno sprite voluto, così da poter vedere alternativamente tutti i 31 sprite. Viene chiesto: "Figure A-G?" cioè in quale figura volete far comparire il disegno, e poi "View Sprite (1-31)?" cioè quale sprite volete vedere. Lo sprite, cioè il suo blocchetto di dati, resta li dov'è, non viene spostato in memoria; è la figura scelta che visualizza un nuovo blocco di dati; la scritta sotto la figura indica sempre quale sprite è visualizzato in quel momento.

C (Color): permette di colorare una figura. Ogni figura, in modo normale, ha due colori: quello di fondo (nel nostro caso il grigio) e un colore definibile dall'utente che può sceglierlo tra i 16 possibili illustrati dai pallini sulla sinistra. Il programma



Il Jacksoniano ha il Basic

Video Basic, corso su cassetta per parlare subito

Oggi è davvero facile imparare il Basic, con Video Basic il corso su cassetta che ti permette di programmare subito il tuo computer. È facile: tu chiedi, lui risponde, tu impari.

Passo dopo passo. Sul tuo schermo appaiono le domande, le risposte, gli esercizi e tu, senza fatica, presto e bene, impari a dialogare col tuo computer, sia un VIC 20, un Commodore 64 o un Sinclair
Video Basic è in edicola. Provalo subito.

Oggi il Basic si impara così.



VIDEO BASIC
20 VIDEOLEZIONI DI BASIC PER IMPARARE COL VIC 20
GRUPPO EDITORIALE JACKSON
1
COMMODORE VIC 20
C-64
MODORE 64
Spectrum 16K/48K/PLUS
SUPER
Versione VIC 20
J.soft
facile
in mano.
col tuo computer.
GRUPPO EDITORIALE JACKSON
IN EDICOLA DALL' 8·1·'85
In omaggio
una fantastica cassetta giochi.
GRUPPO ETHOS

Listato 1. *Il programma Sprite Editor.*

```
1 REM--------SPRITE EDITOR------------
2 REM
3 REM--------CHIARA TOVENA------------
4 REM
5 REM---------15.11.1984--------------
6 REM
7 REM
8 IFPEEK(44)<>16THENPRINT"[<2CRSR D>]EHI,
 I PUNTATORI!":PRINT"POKE44,16:POKE4096,0
":END
9 REM
10 GOTO10007:REM MAIN PRGM
11 REM
95 REM
96 REM
97 REM--------ESPANS. SPRITE-----------
98 REM
110 PRINT"[<1CLR>][<1HOME>]";
114 ES$="."
116 PRINT"  0 [<1CHR$(165)>]        [<1CHR
$(165)>]         [<1CHR$(165)>]        [<1CH
R$(165)>]"
120 FORI=1TO21:IFI<10THENPRINT" ";
130 PRINTI:NEXT
140 GOSUB200:RETURN
150 REM
180 REM-----------RETICOLO------------
190 REM
200 M$="":FORI=1TO24:M$=M$+ES$:NEXT
210 PRINT"[<1HOME>]":FORI=1TO21:PRINTTAB(
4)M$:NEXT
220 RETURN
230 REM
240 REM
300 REM
310 REM
320 REM-----------INPUT NUM.-----------
340 REM
350 POKECV,23:POKECH,0:SYSCP:PRINTCA$
355 SYSCP:PRINTDO$;:SS$=""
357 INPUTSS$:CR=SS$=""
358 IFCRGOTO370
360 SP=VAL(SS$):IFSP<LIORSP>LSTHEN350
370 RETURN
380 REM
390 REM
400 REM---------GET ALF.---------------
410 REM
430 CU$="[<1CHR$(166)>][<1CRSR L>] [<1CRS
R L>]"
431 POKECV,23:POKECH,0:SYSCP:PRINTCA$
435 SYSCP:PRINTDO$;
439 REM
440 REM-ENTRY SENZA POS.CUR--
441 L=3:SP$=""
442 T=0
443 GETSP$:IFSP$<>""THEN461
444 T=T+1:IFT< 8GOTO443
445 L=4-L:PRINTMID$(CU$,L,2);
446 GOTO442
460 REM
461 CR=SP$=CHR$(13):RETURN
470 REM
500 REM
510 REM
520 REM-------POSIZ. CURSORE-----------
530 REM
540 CP=820:REM OBJ
550 FORI=0TO7:READA:POKECP+I,A:NEXT
565 CV=CP+1:CH=CP+3
570 RETURN
580 REM
590 REM
595 REM
```

chiede prima quale figura si vuole colorare, poi con che colore; e se deve apparire in modo normale o multicolor. Battendo <Cr> a una domanda come al solito si esce dal comando.

M (Multicolor): se la figura è in multicolor allora può avere altri due colori che sono gli stessi per tutte le figure, e si inizializzano con questo comando. I due colori rappresentano rispettivamente il multicolor reg. # 0 e reg. # 1.

Per facilitare all'utente la creazione di uno sprite multicolor, sotto le figure compare il disegno punto-pallino (che corrisponde al reg. # 0) col suo colore scelto, e il disegno pallino-pallino (corrispondente al reg. # 1) col suo colore; (ricordiamo che punto-punto compare come colore di fondo, e pallino-punto è il colore particolare della figura). Battendo <Cr> si salta alla domanda seguente senza alterare il registro.

S (Save): per salvare uno o un gruppo di sprite su nastro. Il programma chiede "Save from Sprite (1-31)?" cioé da quale sprite si vuol cominciare a salvare e "To Sprite?" cioè fin dove. Ad esempio, per salvare gli sprite 5, 6, 7 e 9, rispondete 5 alla prima domanda e 9 alla seconda (naturalmente salverete anche la 8 per cui è meglio mettere sempre adiacenti gli sprite relativi ad un determinato gioco). Per uscire dalla routine sempre <Cr>; il blocco delle sprite selezionate viene registrato come un tutt'uno e sarà ricaricato sempre in blocco.

L (Load): per ricaricare un blocco di sprite precedentemente registrato. Viene chiesto "Load from Sprite?" cioè a partire da quale sprite bisogna mettere i dati che entreranno. Sul nastro, prima dei dati, è inciso anche il numero degli sprite che il file contiene, per cui il programma lettore può sapere se l'intero blocco ci starà o no in memoria. Ad es. se pretendete di caricare un blocco di 10 sprite partendo a metterlo dalla 25-esima, avrete in risposta un "Not enough space" che vi avverte di caricare gli sprite in un posto precedente; se le mettete a partire dal numero 5, si installeranno appunto dalla 5 alla 14.

Q (Quit): quando avete finito di gio-

VIDEO BASIC
abbonarsi conviene

(5 splendidi raccoglitori insieme al corso completo)



Video Basic lo trovi in edicola a lire 8.000 il fascicolo con cassetta e manuale. Ma abbonarsi conviene; con 165.000 lire avrai infatti il corso completo, a casa tua, e 5 splendidi (e pratici) raccoglitori del valore di 40.000 lire.
NON PERDERE L'OCCASIONE!

Desidero abbonarmi a Video Basic

- ☐ Per il computer Commodore VIC 20
- ☐ Per il computer Commodore 64
- ☐ Per il computer Sinclair Spectrum

Spedire a:
JACKSON
Via Rosellini, 12
20124 Milano

Allego lire 165.000 con assegno n° ______ della Banca ______ o allego fotocopia della ricevuta di versamento con vaglia postale intestato a GRUPPO EDITORIALE JACKSON - MILANO, che mi dà diritto di ricevere a casa mia il corso completo e 5 raccoglitori.

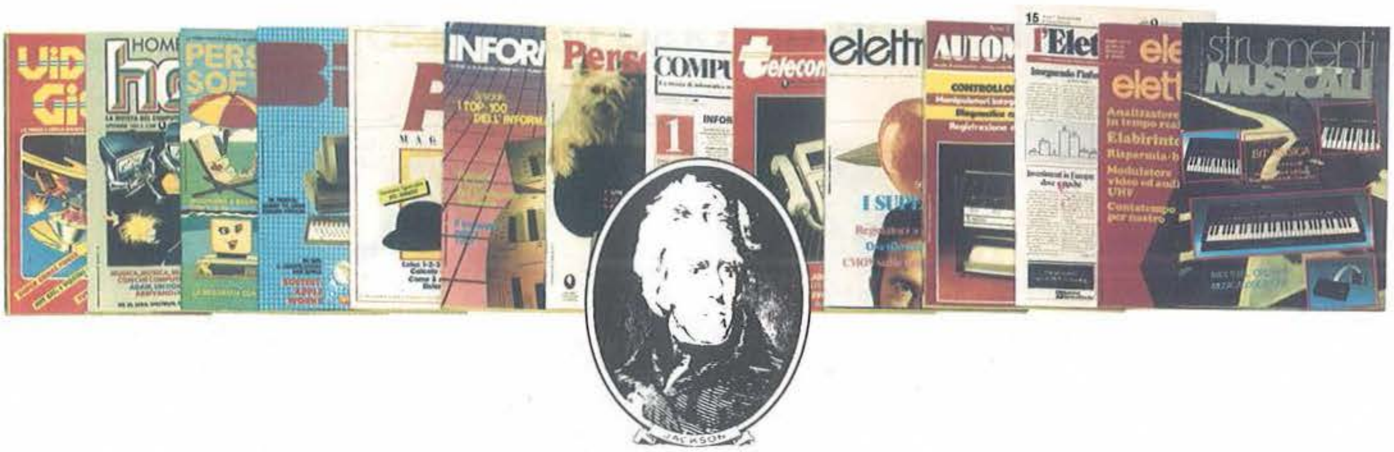
Nome ______ Cognome ______

Via ______ N. ______

CAP ______ Città ______ Provincia ______

Il Jacksoniano sceglie tra 14 top...

Jackson & Sons



Jackson: una grande, esauriente scelta di periodici per sapere tutto ciò che è indispensabile.
In più abbonandoti a queste riviste puoi moltiplicare le tue possibilità di vincere il favoloso premio del grande concorso Jackson.

Videogiochi, la guida indiscussa al fantastico mondo dei videogames;

Home Computer, la rivista del computer in casa;

Personal Software, la rivista dedicata al software dei personal computer;

Bit, la prima rivista europea di personal computer, software, accessori, la più prestigiosa e più diffusa in Italia;

Informatica Oggi, il punto di riferimento obbligato per chi si occupa di sistemi EDP e di Office Automation;

PC Magazine, la prima rivista italiana dei sistemi MS-DOS, Personal Computer IBM e compatibili;
Personal O, la rivista indipendente per gli utenti di PC Olivetti;
Compuscuola, la rivista di informatica nella didattica, per la scuola italiana;
Telecomunicazioni Oggi, la rivista di telecomunicazioni e telematica;
Automazione Oggi, il mensile della nuova automazione industriale;
Elettronica Oggi, la più autorevole rivista di elettronica professionale, strumentazione e componenti;
L'Elettronica, il quindicinale di politica industriale, componentistica, informatica e telecomunicazioni;
Elektor, la più diffusa rivista europea di applicazioni e progettazione elettronica.
Strumenti musicali, il periodico di strumenti musicali e computer-music.

...e ha una biblioteca ricchissima tutta per lui.

(con uno sconto del 20%)*

Richiedete il catalogo inviando lire 3000 in francobolli a:

GRUPPO EDITORIALE JACKSON

Via Rosellini, 12 - 20124 Milano

*Offerta valida solo fino al 28-2-85

Ecco come ti abboni, risparmi, vinci.

ETHOS

VINCI 100 COMMODORE 64

Abbonatevi subito: tra tutti coloro che si abboneranno a una o più riviste Jackson tra il 15/9/84 e il 28/2/85 saranno estratti a sorte mensilmente 20 Commodore 64.



Per sottoscrivere abbonamenti potrete utilizzare il modulo di cc/p inserito in questo fascicolo o inviare un assegno allegato al tagliando sottostante.

Gruppo Editoriale Jackson S.r.l. - Via Rossellini, 12 - 20124 Milano, allegando assegno o fotocopia della ricevuta di versamento con vaglia postale intestato a GRUPPO EDITORIALE JACKSON - MILANO.

☐ Si, desidero sottoscrivere un abbonamento a:

☐ Videogiochi (11 n.) L. 30.000 anzichè L. 38.500
☐ Home Computer (11 n.) L. 31.500 anzichè L. 38.500
☐ Personal Software (11 n.) L. 34.000 anzichè L. 44.000
☐ Bit (11 n.) L. 43.000 anzichè L. 55.000
☐ Informatica Oggi (11 n.) L. 31.000 anzichè L. 38.500
☐ PC Magazine (10 n.) L. 40.000 anzichè L. 50.000
☐ Personal O (10 n.) L. 35.000 anzichè L. 40.000
☐ Compuscuola (9 n.) L. 15.000 anzichè L. 18.000
☐ Telecomunicazioni Oggi (10 n.) L. 28.000 anzichè L. 35.000
☐ Automazione Oggi (11 n.) L. 30.500 anzichè L. 38.500
☐ Elettronica Oggi (11 n.) L. 36.000 anzichè L. 44.000
☐ L'Elettronica (22 n.) L. 44.000
☐ Elektor (12 n.) L. 29.000 anzichè L. 36.000
☐ Strumenti Musicali (10 n.) L. 24.000 anzichè L. 30.000

Attenzione per abbonamento all'estero le tariffe devono essere aumentate del 50%

E c'è un super-risparmio a chi si abbona a due o più riviste.

Tutti coloro che sottoscrivono l'abbonamento a due o più riviste godono di un prezzo ulteriormente agevolato, come appare nella seguente tabellina.
Esempio: Bit+Informatica Oggi L. 43.000+31.000=74.000 meno L. 2.000=L. 72.000

Abbonamento	a 8 riviste: L. 20.000 in meno sulla somma degli 8 prezzi d'abbonamento
a 2 riviste L. 2.000 in meno sulla somma dei 2 prezzi d'abbonamento	a 9 riviste L. 25.000 in meno sulla somma dei 9 prezzi d'abbonamento
a 3 riviste L. 4.000 in meno sulla somma dei 3 prezzi d'abbonamento	a 10 riviste L. 30.000 in meno sulla somma dei 10 prezzi d'abbonamento
a 4 riviste L. 7.000 in meno sulla somma dei 4 prezzi d'abbonamento	a 11 riviste L. 35.000 in meno sulla somma degli 11 prezzi d'abbonamento
a 5 riviste L. 10.000 in meno sulla somma dei 5 prezzi d'abbonamento	a 12 riviste L. 40.000 in meno sulla somma dei 12 prezzi d'abbonamento
a 6 riviste L. 13.000 in meno sulla somma dei 6 prezzi d'abbonamento	a 13 riviste L. 44.500 in meno sulla somma dei 13 prezzi d'abbonamento
a 7 riviste L. 16.000 in meno sulla somma dei 7 prezzi d'abbonamento	a 14 riviste L. 50.000 in meno sulla somma dei 14 prezzi d'abbonamento

☐ Allego assegno n° ______________ della Banca ______________________
☐ Ho effettuato versamento con vaglia postale e allego fotocopia della ricevuta

Nome ______________________
Cognome ______________________
Azienda ______________________
CAP. ______________ Città ______________________
Via ______________________

GRUPPO EDITORIALE JACKSON

cherellare o non ne potete più di quegli stupidi disegnini. Questo comando non ha mai inconvenienti!!! (Anche perché in caso di un tardivo ripensamento potete recuperare intatta la situazione con un Goto 10010, entrata speciale per i distratti. Il programma si presenta come se fosse stato normalmente fatto partire con Run, ma conserva in memoria i vecchi sprite che altrimenti verrebbero cancellati).

10000-11030 Main program (programma principale)

Esegue una subroutine di preparazione di tutto il video e poi va al menu principale. La seconda entrata, chiamata "entrata calda", è un'entrata di emergenza (non dovrebbe mai servire), ma se avete combinato qualche guaio, ad esempio un Run/Stop-Restore, o vi è stato lo scrolling del video, e non avevate salvato il vostro precedente lavoro, potete recuperarlo eseguendo immediatamente un Goto 10010 prima di fare qualsiasi altra cosa: si reinizializza tutto mantenendo però in memoria i dati dei vecchi sprite. La routine del menu non necessita di spiegazioni, notate solo nella Quit (linee 11012-11030) che per sicurezza viene chiesto di nuovo se si vuole veramente smettere. Da ultimo gli sprite sono spenti, ritornano i colori standard del video, e tutto è pronto per un nuovo gioco.

20005-20500 Inizializzazioni

20010 - La Cold Init (entrata fredda) esegue in più solo un azzeramento di tutti i byte dei 32 blocchetti degli sprite.
20024 - Stabilisce i colori di video, bordo, e caratteri: se per qualunque ragione volete cambiarli fatelo qui.
20037 - Crea una stringa di cancellazione lunga 37 spazi.
20042 - Crea l'array PO() delle potenze di 2: PO(0)=1, PO(1)=2 ... PO(7)=128.
20185 - Inizializza l'indirizzo del controller e della memoria dove si trovano i byte degli sprite.
20190 - Sistema la posizione sul video delle 8 figure accese: i valori delle coordinate X e Y sono letti dai

Seguito listato Sprite Editor.

```
600 REM----SUB.DI POSIZIONAMENTO-------
605 REM-------E CANCELLAZIONE----------
610 REM
620 POKECV,23:REM 1 ENTRY
625 POKECH,0:SYSCP:REM 2 ENTRY
630 PRINTCA$:SYSCP
640 RETURN
650 REM
1000 REM
1010 REM
1020 REM--------EDIT SPRITE------------
1022 REM
1025 DO$="EDIT MODE (Q TO QUIT).": GOSUB6
20:PRINTDO$
1030 REM
1040 RI=0:CO=0
1045 POKECV,BV+RI:POKECH,BH+CO:SYSCP
1052 B1=INT(CO/8):B2=7-8*(CO/8-B1):B=DS+R
I*3+B1:REM IND BYTE
1053 PB=PEEK(B):P2=PO(B2)
1054 S$=".":IFPBANDP2THENS$="[<1CHR$(209)
>]"
1055 PRINTS$"[<1CRSR L>]";:REM PER ACCELE
RARE
1056 CU$=" [<1CRSR L>]"+S$+"[<1CRSR L>]":
REM PUNTO O PALLINO
1058 GOSUB441:A$=SP$:REM GET
1059 PRINTS$"[<1CRSR L>]";
1060 IFA$="Z"THENRI=RI+1:IFRI>20THENRI=0
1070 IFA$="A"THENRI=RI-1:IFRI<0 THENRI=20
1080 IFA$="[<1CRSR R>]"THENCO=CO+1:IFCO>2
3THENCO=0
1090 IFA$="[<1CRSR D>]"THENCO=CO-1:IFCO<0
THENCO=23
1092 IFA$=" "THENPOKEB,(PBANDNOTP2)OR(NOT
PBANDP2):REM TOGGLE
```

```
1100 IFA$="Q"THENRETURN
1110 GOTO1045:REM ALTRO TASTO
1300 REM
1310 REM
1320 REM
2000 REM
2010 REM
2020 REM---------PUT(WRITE)------------
2030 REM
2040 DO$="PUT IN SPRITE (1-31) ":LI=1:LS=
31:GOSUB350
2045 IFCRTHEN2060
2050 Q=DS+SP*64:FORI=0TO62:POKEQ+I,PEEK(D
S+I):NEXT
2060 RETURN
2070 REM
2080 REM
3000 REM
3010 REM
3020 REM------------COLOR---------------
3030 REM
3040 DO$="FIGURE (A-G) ? ":GOSUB430
3045 IFCRTHEN3130
3050 IFSP$<"A"ORSP$>"G"THEN3040
3060 SA=ASC(SP$)-64
3070 DO$="COLOR N.(0-15) ":LI=0:LS=15:GOS
UB350
3075 IFCRTHEN3130
3080 POKEV+39+SA,SP:REM COLORA
3090 DO$="<N>ORMAL OR <M>ULTICOLOR ? ":GO
SUB430:K=PEEK(V+28)
3100 IFSP$="N"THENPOKEV+28,KANDNOTPO(SA):
GOTO3130
3110 IFSP$="M"THENPOKEV+28,KORPO(SA):GOTO
3130
3115 IFCRGOTO3130
3120 GOTO3090
3130 RETURN
```

Sprite Editor

Seguito listato Sprite Editor.

```
3140 REM
4000 REM
4010 REM------------SAVE---------------
4020 REM
4030 DO$="SAVE FROM SPRITE (1-31) ":LI=1:
LS=31:GOSUB350:A=SP
4035 IFCRTHEN4120
4040 DO$="TO SPRITE (1-31) ":LI=1:LS=31:G
OSUB350:B=SP
4045 IFCRTHEN4120
4050 NS=B+1-A:IFNS<1THEN4030
4060 GOSUB620:POKECV,22:SYSCP:REM POS
4065 POKEV+21,0:REM SPEGNE SPR
4070 OPEN1,1,1,"SPRITE.DATA"
4090 PRINT#1,NS:IS=DS+A*64:FS=DS+B*64+63
4100 FORI=ISTOFS:PRINT#1,PEEK(I):NEXT:CLO
SE1
4105 POKEV+21,255:REM RIACCENDE
4110 POKECV,22:GOSUB625
4120 RETURN
4200 REM
4998 REM
4999 REM
5000 REM---------GET(READ)-------------
5001 REM
5010 IFQQ$<>"G"THEN5022
5015 DO$="GET SPRITE (1-31) ":LI=1:LS=31:
GOSUB350
5020 IFCRTHEN5070
5022 POKECH,4:POKECV,1:SYSCP:Z=0
5030 Q=DS+SP*64:FORI=0TO62:K=PEEK(Q+I):PO
KEDS+I,K
5040 FORJ=7TO0STEP-1:A$=".":IFK>=PO(J)THE
NA$="[<1CHR$(209)>]":K=K-PO(J)
5042 PRINTA$;:NEXTJ
5050 Z=Z+1:IFZ>2THENZ=0:PRINT:PRINT"[<4CR
SR R>]";
5060 NEXTI
5070 RETURN
6000 REM
6010 REM
6015 REM
6030 REM------------VIEW--------------
6040 REM
6050 DO$="FIGURE (A-G) ? ":GOSUB430
6055 IFCRTHEN6125
6060 IFSP$<"A"ORSP$>"G"THEN6050
6070 SA=ASC(SP$)-64
6080 DO$="VIEW SPRITE (1-31) ":LI=1:LS=31
:GOSUB350
6085 IFCRTHEN6125
6090 POKE2040+SA,32+SP
6100 REM-AGGIORNAMENTO N.SPR
6110 POKECH,XS(SA)+2:POKECV,YS(SA):SYSCP
6120 PRINTMID$(STR$(SP)+" ",2)
6125 RETURN
6130 REM
6140 REM
7000 REM
7010 REM
7030 REM-----------MULTICOLOR---------
7040 REM
7060 DO$="MULTICOLOR REG.#0 (0-15) ":LI=0
:LS=15:GOSUB350
7065 IFCRTHEN7090
7070 POKEV+37,SP
7080 POKECH,XS(8)+3:POKECV,YS(8):SYSCP:PR
INT"[<1CHR$(209)>]":POKE56168,SP
7090 DO$="MULTICOLOR REG.#1 (0-15) ":LI=0
:LS=15:GOSUB350
7095 IFCRTHEN7120
7100 POKEV+38,SP
7110 POKECH,XS(9)+3:POKECV,YS(9):SYSCP:PR
INT"[<1CHR$(209)>]":POKE56174,SP
7120 RETURN
8000 REM
8005 REM
8010 REM
8030 REM------------NEW---------------
8040 REM
8060 DO$="EDIT FIGURE: <E>MPTY OR <F>ILL
? ":GOSUB430
```

Data (linea 21010), mentre il byte più significativo di X è posto a 1 per tutte perché le figure si trovano nella parte destra del video (quindi con X maggiore di 255).
20205 - Pone il bianco come colore iniziale delle figure: se non vi va bene cambiatelo qui.
20208 - Sistema i puntatori delle figure, segnalando che sono posti al 32°, 33° ... ecc. blocchetto di 64 byte.
20250 - Legge dai Data (linee 21020-21030) la posizione orizzontale e poi verticale delle scritte sotto le figure e la posizione dei segnalatori del multicolor.
20260 - Si posiziona giusto e scrive "Edit" sotto la figura 0.
20270 - Per le altre figure un ciclo For-Next scrive la lettera corrispondente (individuata dal carattere ASCII pari a 64+I) seguita da = e dal numero dello sprite che la figura visualizza.
20310 - Disegna in colonna 16 pallini e li colora. La locazione MC è la mappa di memoria dei colori, e "pokando" in essa dei valori le corrispondenti caselle del video appaiono del colore scelto. La casella MC è la prima del video, la MC+40 è la prima della seconda riga e così via per 16 righe.
20330 - Nelle posizioni adeguate vengono accesi i segnalatori del multicolor. I colori saranno accesi solo dall'utente con l'apposito comando M del menu.
21000 - Seguono in ordine tutti i Data del programma.
E con questo bel popo' di cose da fare non dovete meravigliarvi se ci mette tanto tempo a partire ...

110-220 Costruzione del reticolo

Cancella il video e si posiziona ad Home: stampa i segnalini separatori dei 3 byte di cui è composta ogni linea dello sprite, poi si occupa dei numeri delle colonne da far comparire sulla sinistra del video, che servono a facilitare il centraggio del disegno e a ricordare il numero dei colori. I numeri sono stampati giustificati a destra (linee 120-140). La subroutine 200 (Reticolo) stampa le 21 file di puntini dopo aver creato

Sprite Editor

Seguito listato Sprite Editor.

```
8065 IFCRTHEN8115
8070 IFSP$<>"E"ANDSP$<>"F"THENGOTO8060
8080 N=0:ES$=".":IFSP$="F"THENN=255:ES$="
[<1CHR$(209)>]"
8100 FORI=0TO62:POKEDS+I,N:NEXT
8110 GOSUB200:REM RETICOLO
8115 RETURN
8120 REM
8130 REM
9000 REM
9010 REM------------LOAD--------------
9020 REM
9030 DO$="LOAD FROM SPRITE (1-31) ":LI=1:
LS=31:GOSUB350:A=SP
9035 IFCRTHEN9100
9040 GOSUB620:POKECV,21:SYSCP
9045 POKEV+21,0:REM SPEGNE SPR
9050 OPEN1,1,0,"SPRITE.DATA"
9060 INPUT#1,NS:B=A+NS-1
9070 IFB>31THENCLOSE1:GOTO9300
9080 IS=DS+A*64:FS=DS+B*64+63
9090 FORI=ISTOFS:INPUT#1,C:POKEI,C:NEXT:C
LOSE1
9094 POKECV,22:GOSUB625
9095 POKEV+21,255
9100 RETURN
9105 REM
9300 POKECV,22:GOSUB625:GOSUB620
9301 PRINT"NOT ENOUGH SPACE FOR"NS"SPRITE
"
9305 FORI=1TO5000:NEXT
9310 GOTO9095
9320 REM
9340 REM
9500 REM
9510 REM-----------DATA---------------
9520 REM
9530 REM
9540 DO$="DATA OF SPRITE (1-31) ":LI=1:LS
=31:GOSUB350
9550 IFCRTHEN9630
9555 ES$=".":GOSUB200
9560 IS=DS+SP*64:T(0)=4:T(1)=12:T(2)=20
9570 PRINT"[<1HOME>]":FORI=ISTOIS+62STEP3
9580 FORJ=0TO2:PRINTTAB(T(J))PEEK(I+J);:N
EXTJ
9590 PRINT:NEXTI
9600 DO$="PRESS RETURN TO CONTINUE ":GOSU
B430
9610 IFNOTCRTHEN9600
9620 SP=0:GOSUB5010
9630 RETURN
10000 REM
10001 REM
10002 REM
10004 REM--------MAIN PROGRAM----------
10005 REM
10006 REM
10007 GOSUB20005:REM ENTRATA FREDDA
10008 GOTO10050
10009 REM
10010 GOSUB20023:REM ENTRATA CALDA
10015 REM
10016 REM
10020 REM---------MAIN MENU'----------
10030 REM
10050 DO$="E,G,P,D,V,C,M,N,L,S,Q ? "
10060 GOSUB430:QQ$=SP$:REM GET ALF
10070 IFQQ$="G"THENGOSUB5010:REM GET
10080 IFQQ$="E"THENGOSUB1025:REM EDIT
10085 IFQQ$="P"THENGOSUB2040:REM PUT
10090 IFQQ$="S"THENGOSUB4030:REM SAVE
10100 IFQQ$="L"THENGOSUB9030:REM LOAD
10105 IFQQ$="N"THENGOSUB8060:REM NEW
10110 IFQQ$="V"THENGOSUB6050:REM VIEW
10114 IFQQ$="C"THENGOSUB3040:REM COLOR
10116 IFQQ$="M"THENGOSUB7060:REM MULTIC
10117 IFQQ$="D"THENGOSUB9540:REM DATA
10118 IFQQ$="Q"THEN11000:REM QUIT
10120 GOTO10050
10130 REM
```

una stringa composta da 24 elementi; notate che ES$ (il puntino del reticolo) era stata inizializzata in precedenza (linea 114) e che entrando nella subroutine alla linea 116 è possibile passare ad ES$ un qualsiasi altro carattere, e ottenere così reticoli diversi (la possibilità sarà sfruttata dal comando New).

350-370 Input numerico

Fa entrare da tastiera una linea che sia un numero e chiuda con <Cr>. Le devono essere passati vari parametri: la stringa DO$ con la domanda da porre, e i limiti inferiore e superiore del numero che deve entrare (LI, LS). Se l'input è fuori dai limiti viene scartato e la domanda viene riproposta: la 350 cancella le scritte precedenti stampandovi sopra una stringa di spazi, la 355 pone la domanda e annulla la stringa di input: questo perché, qualora si battesse un solo <Cr>, la stringa di input non verrebbe toccata, e continuerebbe pertanto a contenere il valore non voluto dell'input precedente. La 357 effettua l'input: per non provocare errori viene accettata una stringa di cui, più tardi, sarà calcolato il valore numerico (linea 360). Notate l'espressione CR=SP$="": è una assegnazione "logica": alla variabile CR è assegnato il valore -1 se l'espressione SS$="" è vera, altrimenti CR vale 0. Così se l'input è consistito in un solo <Cr>, la variabile SS$ è rimasta una stringa nulla e CR vale -1.

Considerate infatti questo particolare: quando si esegue la funzione Val di una stringa non numerica si ottiene come risultato 0, esattamente come se fosse stato realmente battuto uno zero. Anche il Val di una stringa nulla è 0, e poiché volevamo distinguere i due casi abbiamo dovuto effettuare questo ulteriore test sulla stringa entrata, prima di convertirla in numero.

430-461 Get alfabetico con cursore

Subroutine con un trucchetto interessante: avere una Get con la presenza rassicurante del cursore.

Sprite Editor

Seguito listato Sprite Editor.

```
10140 REM
10150 REM
11000 REM-----------QUIT--------------
11010 REM
11012 DO$="ARE YOU SURE (Y/N) ? ":GOSUB43
0
11014 IFSP$<>"Y"THENGOTO10050:REM REENT.
11020 POKEV+21,0:POKE53280,14:POKE53281,6
11030 PRINT"[<1AZZ>][<1CLR>]":END
11040 REM
11050 REM
11060 REM
20000 REM
20002 REM---------COLD INIT------------
20003 REM
20004 REM
20005 DS=2048
20009 REM INIT DATA SPRITE
20010 FORI=DSTODS+64*32-1:POKEI,0:NEXT
20014 REM
20015 REM
20020 REM---------WARM INIT------------
20021 REM
20022 REM
20023 RESTORE
20024 POKE53280,11:POKE53281,12:PRINT"[<1
BLU>]":REM SCREEN
20035 GOSUB540:REM POS CURS
20037 CA$="":FORI=1TO37:CA$=CA$+" ":NEXT
20038 BV=1:BH=4
20042 A=1:FORI=0TO7:PO(I)=A:A=A*2:NEXT
20043 REM
20044 REM
20150 GOSUB110:REM ACCENS RETIC
20160 REM
20170 REM ACCENS.SPR 0-7
20180 REM
20185 V=53248:DS=2048
20190 FORI=VTOV+15:READA:POKEI,A:NEXT:POK
EV+16,255:REM POS.X,Y
20205 FORI=V+39TOV+46:POKEI,1:NEXT:REM CO
L.WHITE
20206 POKEV+28,0:REM NON MULTICOL.
20208 FORI=0TO7:POKE2040+I,32+I:NEXT:REM
POINTER SPRITE
20210 POKEV+21,255:REM ACCENS.
20230 REM
20240 REM-SCRITTE SOTTO SPRITE
20242 REM
20250 FORI=0TO9:READXS(I):NEXT
20255 FORI=0TO9:READYS(I):NEXT
20260 POKECV,YS(0):POKECH,XS(0):SYSCP:PRI
NT"EDIT"
20270 FORI=1TO7:POKECV,YS(I):POKECH,XS(I)
:SYSCP
20280 PRINTCHR$(64+I)"="MID$(STR$(I),2):N
EXT
20300 REM
20305 REM 16 PALLINI COLORATI
20310 PRINT"[<1HOME>]";:FORI=0TO15:PRINT"
[<1CHR$(209)>]":NEXT
20320 MC=55296:FORI=0TO15:POKEMC+40*I,I:N
EXT
20325 REM
20326 REM SEGNAL MULTICOL
20330 POKECH,XS(8):POKECV,YS(8):SYSCP:PRI
NT".[<1CHR$(209)>]="
20340 POKECH,XS(9):POKECV,YS(9):SYSCP:PRI
NT"[<2CHR$(209)>]="
20500 RETURN
20510 REM
20520 REM
20530 REM
21000 DATA162,0,160,0,24,76,240,255:REM P
OS.CURS.
21010 DATA7,58,48,58,7,97,48,97,7,137,48,
137,7,178,48,178:REM POS.SPR.
21020 DATA30,35,30,35,30,35,30,35,29,35:R
EM POS.H SCRITTE
21030 DATA4,4,9,9,14,14,19,19,21,21:REM P
OS.V SCRITTE
```

Creiamo una stringa CU$, formata da 4 caratteri: un cursore, un carattere di spazio-indietro, uno spazio bianco e un altro spazio-indietro; la useremo in seguito. Come al solito cancelliamo la linea sottostante e poniamo la nuova domanda DO$ (preparata dalla routine chiamante), poi annulliamo la stringa di input SP$ (431-435). Cosa succede poi? Viene eseguita una Get (443), e se non era stato premuto un tasto incrementa il contatore T: questo per otto volte, dopo di che vengono stampati due caratteri della stringa CU$, i primi due e precisamente il simbolo del cursore e un backspace. In poche parole stampa il cursore. Poi il ciclo riprende con altre otto Get e se non viene premuto nessun tasto in questo tempo il valore di L al secondo passaggio sarà 3 e provocherà la stampa del terzo carattere di CU$, cioè lo spazio, esattamente sopra al cursore stampato prima, poi del backspace. E questo cancella il cursore. Ai passaggi successivi sarà sempre stampato alternativamente un cursore e uno spazio, dando così l'impressione che ci sia un vero cursore lampeggiante. Il gioco finisce quando viene premuto un tasto, per cui si salta alla 461 che pone CR=0 se il carattere entrato non era un <Cr>. SP$ è comunque restituito alla routine chiamante. Potete divertirvi a cambiare la forma del cursore modificando la CU$, o a variare la velocità del lampeggio agendo sulla IF T < 8 (linea 444).

540-570 Posizione cursore

La subroutine legge gli otto Data della linea 21000, che costituiscono un piccolo programma in linguaggio macchina:

```
LDX # 0
LDY # 0
CLC
JMP PLOT
```

La Plot è una subroutine del Kernal, cioè del sistema operativo della Commodore, ed è usabile anche dagli utenti: la sua funzione è di posi-

zionare il cursore in orizzontale e verticale a seconda dei parametri con cui sono inizializzati rispettivamente i registri Y e X. Se ad esempio, Y vale 5 e X 3, il cursore apparirà sul video nella sesta casella della quarta riga (la prima riga e colonna sono la numero 0). Per ulteriori spiegazioni su questa subroutine vedete il Programmer's Reference Guide del C 64 a pag. 290 (versione inglese). A noi interessa sapere che la routine in linguaggio macchina viene messa all'indirizzo indicato da CP, che CH e CV contengono gli indirizzi in cui "pokare" il numero di riga e di colonna in cui si vuole far apparire il cursore.

620-640 Posizionamento del cursore e cancellazione

La 620 inizializza il cursore alla 24-esima riga e la 625 alla prima colonna; poi chiama la subroutine di posizionamento.
In quel punto stampa la stringa di cancellazione per eliminare le scritte precedenti, e con il seguente Sys si riposiziona a capo, pronto per scrivere una nuova linea (il secondo Sys, infatti, utilizza per CH e CV gli stessi parametri dati precedentemente).

Routine

2040-2060 Put cioè salva lo sprite appena disegnato in uno degli altri 31 posti.

In memoria possono starci fino a 32 disegni di sprite diversi: in questo programma lo sprite 0 è usato solo per l'edit, e bisogna poi spostarlo da un'altra parte per poterlo salvare su nastro, o per tenerlo visualizzato mentre se ne disegnano altri.
La 2040 prepara i parametri per la subroutine di input numerico: la stringa da stampare e i limiti dell'input. Se si batte solo <Cr> si esce dalla routine senza compiere nessun lavoro, altrimenti in Q viene caricato l'indirizzo di dove spostare i 63 byte dello sprite di edit (indirizzo base DS + il numero dello sprite moltiplicato per i 64 byte di lunghezza di ogni sprite) e i byte sono ricopiati (linea 2050). Nello sprite di edit resta comunque il suo contenuto, e se non serve più lo si elimina col comando New.

5010-5070 Get cioè carica nello sprite di edit uno degli altri 31 sprite

Operazione inversa della precedente: chiede quale sprite si vuole editare e se la risposta è solo <Cr> si esce dalla routine. Altrimenti viene calcolato l'indirizzo dello sprite scelto in Q ed esso viene ricopiato byte per byte nello sprite di edit, la n. 0 (linea 5030). Bisogna però anche disegnarlo espanso sul reticolo (linea 5040): A$, che rappresenta rispettivamente o il puntino o il pallino, è selezionato e stampato otto volte, in corrispondenza degli otto bit del byte dello sprite letto, e la scelta avviene confrontando il byte con le potenze di 2 via via decrescenti e sottraendo detta potenza dal byte se essa è presente. Il ciclo che coinvolge Z serve a stampare solo 3 byte su ogni riga, e poi ad andare a capo e riposizionarsi sulla linea seguente del reticolo pronti per i prossimi 3 byte (linea 5050).

8060-8115 New inizializza lo sprite di edit

Chiede se si vuole lo sprite di edit tutto vuoto o tutto pieno e in caso di <Cr> esce dalla routine. L'input deve essere necessariamente o una F o una E altrimenti rifà la domanda (linea 8070). A seconda della risposta prepara N = 0 e ES$ uguale al puntino, oppure N = 255 e ES$ eguale al pallino: N è il numero da scrivere nei byte dello sprite e ES$ è il carattere che apparirà sul reticolo. A questo punto si possono "pokare" e rifare il reticolo (linee 8100 e 8110).

3040-3130 Color ovvero il mondo è migliore se colorato

Dapprima domanda su quale figura si vuole agire: se la risposta non è compresa tra la A e la G rifà la domanda (linea 3050). Poi chiede il colore scelto (ricordate che per comodità dell'utente sono visualizzati dei pallini colorati sulla sinistra dello schermo e i colori corrispondono al numero delle colonne del reticolo scritte proprio di fianco. Quanta comprensione per noi programmatori pigri). In ogni caso battendo solo <Cr> si esce.
La 3060 ha un compito speciale: trasformare la lettera entrata nel numero dello sprite corrispondente: la lettera (da A a G) viene trasformata nel suo codice ASCII (da 65 a 71) da cui viene sottratto 64 di modo che alla figura A corrispondano, nel controller, i registri dello sprite 1 e così via fino alla figura G a cui corrispondono i registri dello sprite 8. È così possibile "pokare" direttamente il colore scelto nel registro (linea 3080). Nella scelta tra normale e multicolor le cose sono un po' più complesse: infatti c'è un solo registro di multicolor che serve per tutti 8 gli sprite (un bit per ogni sprite). Il registro viene letto, poi in caso di Normal si azzera il bit corrispondente alla figura scelta (linea 3100) e in caso di Multicolor il bit viene messo a 1 (linea 3110) usando le istruzioni And e Or logiche.

7060-7120 Multicolor sceglie gli altri due colori

Non c'è molto da dire: pone le domande (linee 7060 e 7090), scrive il colore nel registro corrispondente (linee 7070 e 7100), poi nella posizione adeguata disegna il pallino e lo colora (linee 7080 e 7110). Questi ultimi servono per il disegnatore smemorato, ma sono comunque molto comodi.

6050-6125 View per vedere qualunque sprite

Viene domandato in quale delle 7 figure dovrà apparire lo sprite scelta: poi la 6090 "poka" nel puntatore di detta figura l'indirizzo del nuovo blocchetto di dati. Questo indirizzo è espresso come numero del blocchetto: se i nostri sprite sono posti in memoria all'indirizzo 2048 e ogni sprite occupa 64 byte (63+1 vuoto), allora quell'indirizzo rappresenta il 32° blocchetto di 64 byte; il secondo sprite sarà il 33° blocchetto e così via. Poi aggiorna la scritta sottostante la figura: ci si posiziona alle coordinate dovute (linea 6110) e viene stampato il numero del nuovo sprite. La linea 6120 fa qualcosa di più: stampa il numero senza quel fastidioso spazio che precede la stampa di tutti i numeri positivi: il numero, col suo spazio, è trasformato in stringa (a cui si aggiunge un altro spazio in fondo per cancellare eventuali numeri già stampati sotto), e la stringa è stampata a partire dal secondo carattere, cioè senza lo spazio iniziale.
Simpatico, no?

9540-9630 Data per sapere i byte di uno sprite

Dopo aver chiesto quale sprite, ripulisce il reticolo di edit dove stamperà i numeri (linea 9555). Poi calcola l'indirizzo iniziale IS e le tabulazioni orizzontali di dove dovranno apparire i tre byte su ogni linea (linea 9560). Il seguente For-Next serve a stampare i dati, tre su ogni linea come dicevamo sopra (ciclo che coinvolge J) e ben tabulati sul reticolo. Come esercizio potete imparare a calcolare a mano i byte e poi verificare la loro esattezza ...
Per terminare viene chiesto di premere <Cr>. A questo punto bisogna ricostruire sul reticolo il disegno originario: ma noi abbiamo già una routine che fa questo lavoro, la Get, e la chiamiamo entrando dopo le domande, avendo l'accortezza di porre il numero dello sprite uguale a 0. In questo modo lo sprite di edit viene ricopiato in se stesso e naturalmente anche disegnato sul reticolo.

1025-1110 Edit

Il lavoro è un po' complesso, vediamolo in dettaglio:
2045 - Pone il cursore alla riga e colonna dove si è arrivati. La prima volta sarà nell'angolo in alto a sinistra, poi di volta in volta in una nuova posizione.
1052 - Esegue calcoli sulla attuale posizione del cursore: B1 è il numero del byte sulla riga, B2 è il numero del bit all'interno di B1, B è l'indirizzo del byte corrispondente nella memoria dati, PB il contenuto di B e P2 il valore della potenza di 2 corrispondente a B2.
1054-1055 - Si predispone a far blinkare (lampeggiare) il puntino o il pallino sottostante al cursore: per decidere quale dei due, verifica se il bit corrispondente del byte su cui si trova è a 0 oppure a 1. Lo stampa e riporta indietro il cursore.
1056-1059 - Prepara la nuova stringa cursore CU$, che sarà usata dalla subroutine "Get con cursore". Qui aspetta un carattere battuto da tastiera, poi torna.
1060-1110 - Analizza il carattere entrato: se è Z incrementa il contatore di riga RI (se oltrepassa l'ultima riga viene riportato in cima); se è A decrementa lo stesso contatore (portandolo in basso se era in cima); se era <curs. des> incrementa il contatore di colonna CO (riportandolo sul bordo sinistro se era tutto a destra); e se era <curs. giù> decrementa detto contatore. Se invece era <spazio> deve invertire il bit su cui si trova: il lavoro viene effettuato "pokando" nel byte B quell'incredibile espressione che vedete nel listato (linea 1092) e che cercheremo di spiegarvi con calma: se volete invertire un bit di un numero esadecimale A, basta avere un altro numero B (detto "maschera") con tutti i bit a 0 tranne un 1 proprio nel posto del bit che volete invertire. Applicando ciecamente la formula "(A And Not B) Or (Not A And B) e facendo qualche scongiuro ricavate un numero C uguale all'originale A solo che ha il bit invertito. Provare per credere, ma è meglio che crediate e basta!!! Il nuovo byte scritto in memoria provoca automaticamente l'accensione o lo spegnimento del puntino sulla figura di Edit. Poi si ricomincia da capo, posizionandosi sulla nuova riga o colonna, o visualizzando il nuovo punto o il nuovo pallino.
Se il tasto premuto era Q si esce dalla routine, se era un altro tasto qualsiasi viene ignorato.

4030-4120 Save su cassetta gli sprite disegnati

Anzitutto domanda da quale a quale sprite si vuol salvare (A e B), e NS è il numero complessivo di essi (linea 4050); qualora fosse uguale a 0 o un numero negativo vuol dire che c'è stato un errore nell'input e si richiedono i valori.
Quindi si posiziona più in alto del solito in modo da evitare che le successive scritte provochino uno scrolling del video; la linea 4065 ha la funzione di spegnere gli sprite durante il salvataggio e lo stesso faremo durante il caricamento: perché mai, direte voi? Semplice, perché l'integrato video, che gestisce gli sprite e altre funzioni che rendono il Commodore 64 un computer così interessante, per fare tutti i suoi lavori ha bisogno di tempo, che ruba ai programmi che stanno funzionando in quel momento.
Normalmente però l'utente non se ne accorge, sia perché il tempo "rubato" è piccolo, sia perché in genere i programmi non sono "temporizzati" cioè non devono fare una certa cosa senza essere assolutamente distratti. Ma durante operazioni delicate come il salvataggio e il caricamento da nastro il microprocessore non può essere disturbato e infatti è noto che durante queste fasi il video viene spento. Anche gli sprite attivi portano via tempo per cui in fase di salvataggio i dati verrebbero spediti

al registratore in modo irregolare e impiegandoci più tempo del dovuto. I guai seri sorgono invece in fase di caricamento: se il microprocessore, invece di tener d'occhio costantemente la linea di ingresso dati, dovesse occuparsi anche degli sprite, rischierebbe di perdere il carattere in arrivo con conseguente errore.
Ma torniamo al nostro programma: la linea 4070 apre un file in scrittura su cassetta chiamato "Sprite.Data"; vi stampa per prima cosa il valore di NS, cioè il numero degli sprite che saranno salvati poi in ordine tutti i byte dall'indirizzo iniziale IS al finale FS (linea 4100). Poi il file è chiuso e si possono riaccendere gli sprite. La 4110 cancella le vecchie scritte e il lavoro è terminato.

9030-9310 Load per ricaricare gli sprite già salvati

Inizialmente chiede da dove deve iniziare a mettere il blocco di sprite letti per poter controllare che ci stiano in memoria; poi spegne gli sprite accesi, apre in lettura il file "Sprite.Data" e legge il numero di sprite incisi. Se dai calcoli (linea 9060) risulta che non ci stanno tutti in memoria nella posizione voluta dall'utente lo segnala e termina il comando. Se invece ci stanno allora li carica da IS a FS calcolati come nella linea 9080. Chiude il file, cancella le vecchie scritte, riaccende gli sprite ed esce.
Una volta disegnati gli sprite, come fare ad usarli? Il metodo più semplice è il copiarsi i byte degli sprite e metterli in Data del proprio programma. Al momento opportuno il programma leggerà i Data, mettendoli in determinate caselle di memoria e i disegni saranno così disponibili.
Se gli sprite invece sono molti o volete fare un lavoro più pulito, salvate su cassetta il blocco degli sprite proprio di seguito al vostro programma. Se inserite nel programma le seguenti linee:

```
10 IS=2048
20 OPEN1,1,0 "SPRITE.DATA"
30 INPUT # 1,A:
   FS=IS+64★A—1
40 FOR I=IS TO FS: INPUT
   #1,A: POKEI,A: NEXT
50 CLOSE1
```

Quando il programma parte si caricherà da solo il blocco degli sprite. Tenete bene conto che IS è l'indirizzo di memoria in cui saranno caricati i dati, state perciò attenti che non si sovrappongano al vostro programma.
Sarebbe buona pratica infatti, metterli prima del programma, così si è sicuri che anche modificando o allungando il programma essi non interferiscono. In questo caso fate i Poke iniziali. ■

Spedizione in Abbonamento Postale - gruppo III/70

L. 2.000

COMPUSCUOLA

La rivista di informatica nella didattica per la scuola italiana

GRUPPO EDITORIALE JACKSON

4 E' IN EDICOLA

Un ponte fra industria e scuola

Con una interessante Tavola Rotonda dal titolo: "Nuove tecnologie nella scuola: uno scenario per il futuro nel rapporto scuola lavoro" si è concluso il Seminario tenutosi lo scorso mese presso il Centro Commerciale Americano, in occasione della mostra Didactics USA 1984. Obiettivo dell'iniziativa tentare di gettare un ponte fra industria e scuola, di mettere a confronto i modi di formazione dei quadri del mondo del lavoro e il mondo della scuola e di verificare se l'informatica può essere considerata utile terreno di incontro fra queste due culture.

A Mauro Laeng, Ordinario di Pedagogia all'Università di Roma, padre riconosciuto dell'introduzione dell'informatica nella scuola italiana, è toccato il compito di introdurre i lavori. "Siamo di fronte ad un vertiginoso aumento della quantità di informazione, è questo un compito difficile da fronteggiare. Se consideriamo la produzione di libri e quotidiani degli Stati Uniti ci troviamo di fronte ad alcuni miliardi di bit all'anno con un accorciamento dei tempi di raddoppio sempre maggiore, con un processo quasi esponenziale. La sfida è imponente, implica una redistribuzione dei compiti formativi. Inoltre dobbiamo tener conto del prolungamento della giovinezza, che ormai sfiora i trent'anni, ed un prolungamento dell'età matura. Sarà necessario, quindi fare ricorso ad una multimedialità, con gergo inglese ad una panoplie, attrezzandoci con tutti gli strumenti mediali atti a colpire il bersaglio. I nuovi media avranno capacità di adattamento alle discipline; è singolare quando si analizza il computer accorgersi che è in grado di fare tutto proprio peché dentro non ha niente, salvo la capacità di fare tutto. Il computer è un mezzo trasparente che veicola a ciò di cui è il messaggio o il tramite, la scuola dovrà affrontare questa sfida.

Parini scriveva con la penna d'oca, Hemingway con la macchina da scrivere, e Umberto Eco ci confessa di usare il word processor. Secondo Maria Gallo, insegnante dell'Itis Armellini di Roma, "La scuola non deve dare quantità di informazioni ma qualità, i docenti dovrebbero poter insegnare tecniche e tecnologie, e tentare di gettare le premesse per la riconversione professionale, evitando così molte frustrazioni

A Bologna il 30-31 gennaio e 1 febbraio Palazzo dei Congressi Convegno Internazionale: «Informatica e Nuove Tecnologie per l'Educazione e la Formazione». All'ordine del giorno il progetto ANTEM che promuove la collaborazione internazionale per lo sviluppo e la diffusione delle nuove tecnologie applicate alla didattica. Promotori: Cnr ltd, Csata, Enea, Dioikema, Università di Milano, Istituto di Cibernetica

Il convegno organizzato dal CIDI su informatica e processi formativi nella scuola, previsto per il 17-18-19 gennaio a Sesto San Giovanni, Milano, è stato rinviato *a causa delle condizioni metereologiche* approssimativamente alla metà di marzo.

Il 16-17-18 gennaio organizzato dal Comune di Lugo di Romagna presso l'aula magna del Liceo Scientifico Statale «G. Ricci Curbastro» convegno: Scuola più: più istruzione, più opportunità, in un sistema formativo integrato. Il convegno è promosso con la collaborazione del Dipartimento di Scienze dell'Educazione dell'Università di Bologna, il patrocinio della regione Emilia Romagna, della provincia di Ravenna e del Provveditorato.

Organo elettronico

Trasformate il vostro Sharp in un potente strumento musicale.

di *Martino Sangiorgio*

Con questo programma è possibile utilizzare il computer Sharp MZ-721 (o MZ-731) come un organo elettronico. Infatti, guidati dal video su cui compare il disegno della tastiera, coi tasti bianchi e anche con quelli neri (per i diesis), basta pigiare sui tasti dalla "Q" al "?" per ottenere le note normali, mentre pigiando sui tasti dall'"1" al ":" si otterranno i diesis. Lo Sharp MZ-700, si sa, non ha un generatore di suoni così sofisticato come altri elaboratori, anche di fascia più bassa (esempio CBM 64 e TI99/4A): ha una sola voce, e quindi non si possono generare accordi, copre una gamma di tre sole ottave e, in più, il volume non è modificabile via software.

Il suono che è in grado di generare è però di notevole qualità, e questo programma lo dimostra. La gamma di suoni coperta non è tutta quella possibile con lo Sharp, ma va dal Fa minore (tasto "Q") al Sol maggiore (tasto "?"), per un totale di 27 note musicali diverse.

Buon lavoro, quindi, e ... musica, Maestro!

Il programma

Il programma è in linguaggio macchina, e per la sua digitazione vi sono tre possibilità.

A) Utilizzare, se ne è in possesso, l'Editor-Assembler SP-2102 (è quello dello Sharp MZ80K), o analogo

Listato 1 - *Elenco istruzioni del programma in linguaggio Assembler Z80.*

```
** Z80 ASSEMBLER SP-2102 PAGE 01 **

0000                ;
0000                ;ORGANO ELETTRONICO
0000                ;
0000                MNTR:   EQU   0000H
0000   P            MSTP:   EQU   0047H
0000   P            MSTA:   EQU   0044H
0000   P            GETKY:  EQU   001BH
0000   P            LETNL:  EQU   0006H
0000   P            MSG.    EQU   0015H
0000   P            PRNT:   EQU   0012H
0000                        SKP   3


0000 3E16           START:  LD    A,16H
0002 CD1200                 CALL  PRNT
0005 113501                 LD    DE,MSG1
0008 CD1500                 CALL  MSG
000B CD0600                 CALL  LETNL
000E 115501                 LD    DE,MSG2
0011 CD1500                 CALL  MSG
0014 CD0600                 CALL  LETNL
0017 113501                 LD    DE,MSG1
001A CD1500                 CALL  MSG
001D CD0600                 CALL  LETNL
0020 CD0600                 CALL  LETNL
0023 CD0600                 CALL  LETNL
0026 117501                 LD    DE,MSG3
0029 CD1500                 CALL  MSG
002C CD0600                 CALL  LETNL
002F CD0600                 CALL  LETNL
0032 113B01                 LD    DE,MSG4
0035 CD1500                 CALL  MSG
0038 CD0600                 CALL  LETNL
003B CD1500                 CALL  MSG
003E CD0600                 CALL  LETNL
0041 CD1500                 CALL  MSG
0044 CD0600                 CALL  LETNL
0047 11C201                 LD    DE,MSG5
004A CD1500                 CALL  MSG
004D CD0600                 CALL  LETNL
0050 CD1500                 CALL  MSG
0053 CD0600                 CALL  LETNL
0056 CD1500                 CALL  MSG
0059 CD0600                 CALL  LETNL
005C CD0600                 CALL  LETNL
005F 11E301                 LD    DE,MSG6
0062 CD1500                 CALL  MSG
0065 CD0600                 CALL  LETNL
0068 CD0600                 CALL  LETNL

** Z80 ASSEMBLER SP-2102 PAGE 02 **

006B CD0600                 CALL  LETNL
006E 111002                 LD    DE,MSG7
```

che giri su MZ-700. In questo caso è possibile utilizzare il listato 1, cioè l'elenco delle istruzioni del programma in linguaggio Assembler Z80.

Dopo aver completato la digitazione, si dovrà convertire il programma in linguaggio macchina (per esempio con il "Relocatable loader" SP-2301, o analogo) a partire dalla locazione di memoria $A000. Anche l'indirizzo di esecuzione dovrà essere $A000.

L'utilizzo di questi programmi di sistema è spiegato nei vari manuali, per cui non ci sembra opportuno procedere in questa sede ad una loro sintesi.

B) Utilizzare il Monitor 1Z-013A di sistema, incorporato nei primi 4 Kbyte di memoria, oppure il Monitor presente nella sezione IOCS dell'interprete BASIC.

In entrambi i casi si dovrà usare il listato 2, e si dovrà procedere come segue.

1) Se si vuole usare il Monitor di sistema, premere il pulsante Reset sul retro dell'elaboratore (naturalmente a macchina accesa).

Se invece si vuole usare il Monitor dell'interprete BASIC (il BASIC deve, in questo caso, essere già stato caricato in macchina), digitare il comando Bye e premere Cr. Questo comando passa il controllo al Monitor del BASIC. In entrambi i casi, la procedura seguente è in comune.

2) Digitare, nella posizione del cursore, vicino all'asterisco: MA000 e premere Cr. Questo permette di iniziare la modifica della memoria. Il video presenterà ora la seguente riga:

A000 00

Si dovrà digitare, nella posizione attuale del cursore, il primo numero esadecimale prelevato dal listato 2 (così com'è, senza modificarlo né convertirlo), dopodiché si dovrà

Seguito listato 1.

```
0071 CD1500            CALL  MSG
0074 3E07              LD    A,07H
0076 2119D9            LD    HL,D919H
0079 0625              LD    B,37D
007B 77       DISP1:   LD    (HL),A
007C 23                INC   HL
007D 05                DEC   B
007E C27B00            JP    NZ,DISP1
0081 2141D9            LD    HL,D941H
0084 0625              LD    B,37D
0086 77       DISP2:   LD    (HL),A
0087 23                INC   HL
0088 05                DEC   B
0089 C28600            JP    NZ,DISP2
008C 2169D9            LD    HL,D969H
008F 0625              LD    B,37D
0091 77       DISP3:   LD    (HL),A
0092 23                INC   HL
0093 05                DEC   B
0094 C29100            JP    NZ,DISP3
0097 2191D9            LD    HL,D991H
009A 0625              LD    B,37D
009C 77       DISP4:   LD    (HL),A
009D 23                INC   HL
009E 05                DEC   B
009F C29C00            JP    NZ,DISP4
00A2 21B9D9            LD    HL,D9B9H
00A5 0625              LD    B,37D
00A7 77       DISP5:   LD    (HL),A
00A8 23                INC   HL
00A9 05                DEC   B
00AA C2A700            JP    NZ,DISP5
00AD 21E1D9            LD    HL,D9E1H
00B0 0625              LD    B,37D
00B2 77       DISP6:   LD    (HL),A
00B3 23                INC   HL
00B4 05                DEC   B
00B5 C2B200            JP    NZ,DISP6
00B8 CD4700   START0:  CALL  MSTP
00BB CD1B00   START1:  CALL  GETKY
00BE B7                OR    A
00BF 28F7              JR    Z,START0
00C1 FE21              CP    21H
00C3 CA0000            JP    Z,MNTR
00C6 47                LD    B,A
00C7 21E300            LD    HL,SCTBL
00CA 7E       CMPR:    LD    A,(HL)
00CB FEF0              CP    F0H


   ** Z80 ASSEMBLER SP-2102 PAGE 03 **


00CD 28E3              JR    Z,START0
00CF 23                INC   HL
00D0 B8                CP    B
00D1 2804              JR    Z,+6
00D3 23                INC   HL
00D4 23                INC   HL
00D5 18F3              JR    CMPR
00D7 5E                LD    E,(HL)
00D8 23                INC   HL
00D9 56                LD    D,(HL)
00DA ED53A111          LD    (11A1H),DE
```

Seguito listato 1.

```
00DE CD4400                    CALL  MSTA
00E1 18DB                      JR    START1
00E3                           SKP   3


00E3 51              SCTBL:    DEFB  51H              ;TASTO "Q"
00E4 A42C                      DEFW  2CA4H            ;FA MINORE
00E6 32                        DEFB  32H              ;TASTO "2"
00E7 002A                      DEFW  2A00H            ;FA DIESIS MINORE
00E3 57                        DEFB  57H              ;TASTO "W"
00EA A827                      DEFW  27A8H            ;SOL MINORE
00EC 33                        DEFB  33H              ;TASTO "3"
00ED 8225                      DEFW  2582H            ;SOL DIESIS MINORE
00EF 45                        DEFB  45H              ;TASTO "E"
00F0 3123                      DEFW  2331H            ;LA MINORE
00F2 34                        DEFB  34H              ;TASTO "4"
00F3 8721                      DEFW  2187H            ;LA DIESIS MINORE
00F5 52                        DEFB  52H              ;TASTO "R"
00F6 E31F                      DEFW  1FE3H            ;SI MINORE
00F8 54                        DEFB  54H              ;TASTO "T"
00F9 EE1D                      DEFW  1DEEH            ;DO CENTRALE
00FB 36                        DEFB  36H              ;TASTO "6"
00FC 341C                      DEFW  1C34H            ;DO DIESIS CENTRALE
00FE 53                        DEFB  59H              ;TASTO "Y"
00FF 921A                      DEFW  1A92H            ;PE CENTRALE
0101 37                        DEFB  37H              ;TASTO "7"
0102 1E19                      DEFW  131EH            ;PE DIESIS CENTRALE
0104 55                        DEFB  55H              ;TASTO "U"
0105 BF17                      DEFW  17BFH            ;MI CENTRALE
0107 49                        DEFB  49H              ;TASTO "I"
0108 5216                      DEFW  1652H            ;FA CENTRALE
010A 39                        DEFB  39H              ;TASTO "9"
010B F114                      DEFW  14F1H            ;FA DIESIS CENTRALE
010D 4F                        DEFB  4FH              ;TASTO "O"
010E D413                      DEFW  13D4H            ;SOL CENTRALE
0110 30                        DEFB  30H              ;TASTO "0"
0111 BC12                      DEFW  12BCH            ;SOL DIESIS CENTRALE
0113 50                        DEFB  50H              ;TASTO "P"


   **  Z80 ASSEMBLER SP-2102 PAGE 04  **


0114 3811                      DEFW  1138H            ;LA CENTRALE
0116 2D                        DEFB  2DH              ;TASTO "-"
0117 C310                      DEFW  10C3H            ;LA DIESIS CENTRALE
0119 40                        DEFB  40H              ;TASTO "@"
011A F10F                      DEFW  0FF1H            ;SI CENTRALE
011C 5B                        DEFB  5BH              ;TASTO "["
011D F70E                      DEFW  0EF7H            ;DO MAGGIORE
011F 5C                        DEFB  5CH              ;TASTO "\"
0120 200E                      DEFW  0E20H            ;DO DIESIS MAGGIORE
0122 FC                        DEFB  FCH              ;TASTO "L"
0123 3D0D                      DEFW  0D3DH            ;PE MAGGIORE
0125 4C                        DEFB  4CH              ;TASTO "L"
0126 340C                      DEFW  0C34H            ;PE DIESIS MAGGIORE
0128 2E                        DEFB  2EH              ;TASTO "."
0129 DF0B                      DEFW  0BDFH            ;MI MAGGIORE
012B 2F                        DEFB  2FH              ;TASTO "/"
012C 2D0B                      DEFW  0B2DH            ;FA MAGGIORE
012E 3A                        DEFB  3AH              ;TASTO ":"
012F 7C0A                      DEFW  0A7CH            ;FA DIESIS MAGGIORE
0131 3F                        DEFB  3FH              ;TASTO "?"
0132 EA03                      DEFW  03EAH            ;SOL MAGGIORE
0134 F0                        DEFB  F0H              ;FINE TABELLA NOTE
0135                 ;
0135                 ;LINEE PER STAMPA PIANOFORTE
0135                 ;
0135                 MSG1:     FNT                    ;PRIMA LINEA
0135 20202020                  DEFM  '         ************************'
0139 20202020
013D 202A2A2A
0141 2A2A2A2A
0145 2A2A2A2A
0149 2A2A2A2A
014D 2A2A2A2A
0151 2A2A2A
0154 0D                        DEFB  0DH
0155                 MSG2:     FNT                    ;SECONDA LINEA
0155 20202020                  DEFM  '         * ORGANO ELETTRONICO *'
0159 20202020
015D 202A204F
0161 5247414E
0165 4F20454C
0169 45545452
```

PERSONAL SOFTWARE

Organo elettronico

premere Cr. La modifica verrà accettata e il video presenterà ora il registro successivo (A001).
Si potrà modificare anche questo registro utilizzando il secondo numero esadecimale del listato 2, battendo poi Cr. E così via, fino a modificare direttamente tutti gli indirizzi da $A000 a $A232, prendendo i dati dal listato 2.
L'operazione di modifica della memoria si può interrompere coi tasti Shift e Break premuti contemporaneamente.
3) Terminata la modifica della memoria si può salvare il programma scritto nel seguente modo: mettere dapprima un nastro nel registratore e quindi digitare, sempre vicino all'asterisco:
$A000A232A000
Questo comando permette di salvare lo spezzone di memoria dall'indirizzo $A000 all'indirizzo $A232, e di imporre l'indirizzo di esecuzione in $A000.
4) Se si vuole procedere all'esecuzione immediata del programma, digitare:
JA000 se si è nel Monitor di sistema, oppure:
GA000 se si è nel Monitor del BASIC.
5) Quando si caricherà successivamente questo programma da cassetta, non dovrà essere presente l'interprete BASIC (utilizzare, eventualmente, il tasto Reset). Appena caricato, esso verrà immediatamente posto in esecuzione.
C) Utilizzare un programma BASIC.
È possibile scrivere un programma che effettui tante Poke dall'indirizzo $A000 all'indirizzo $A232, prendendo i dati dal listato 2 (ricordarsi che, su tale listato, i numeri sono esadecimali).
Si possono anche utilizzare delle istruzioni Data.
Prima di effettuare le Poke, ricordarsi che deve essere stato dato il comando (o istruzione):
Limit 9FFF, per limitare l'area di memoria riservata al BASIC. ■

Seguito listato 1.

```
016D 4F4E4943
0171 4F202A
0174 0D                       DEFB  0DH
0175               MSG3:      FNT                          ;TERZA LINEA
0175 20203220                 DEFM  ' 2 3 4  6 7  3 0 -   \ L
0179 20332020
017D 34202020
0181 36202037

   ** Z80 ASSEMBLER SP-2102 PAGE 05  **

0185 20202039
0189 20203020
018D 202D2020
0191 205C2020
0195 4C202020
0199 3A
019A 0D                       DEFB  0DH
019B               MSG4:      FNT                          ;QUARTA LINEA
019B 20C1C8D5                 DEFM  ' C1C8D5C1C8D5C1C8D5FDC1C8D5C1C8D5FDC1C8D
5C1C8D5C1C8D5FDC1C8D5C1C8D5FDC1C8D5'
019F C1C8D5C1
01A3 C8D5FDC1
01A7 C8D5C1C8
01AB D5FDC1C8
01AF D5C1C8D5
01B3 C1C8D5FD
01B7 C1C8D5C1
01BB C8D5FDC1
01BF C8D5
01C1 0D                       DEFB  0DH
01C2               MSG5:      FNT                          ;QUINTA LINEA
01C2 2020FD20                 DEFM  '  FD  FD  FD FD FD  FD FD FD  FD  FD FD
    FD FD FD'
01C6 20FD2020
01CA FD20FD20
01CE FD2020FD
01D2 20FD20FD
01D6 2020FD20
01DA 20FD20FD
01DE 20202020
01E2 20FD20FD
01E6 20FD
01E8 0D                       DEFB  0DH
01E9               MSG6:      FNT                          ;SESTA LINEA
01E9 20512057                 DEFM  ' Q W  E  R T Y  U I O  P  @ [ ↓  . / ?'
01ED 20204520
01F1 20522054
01F5 20592020
01F9 55204920
01FD 4F202050
0201 20204020
0205 5B20FC20
0209 202E202F
020D 203F
020F 0D                       DEFB  0DH
0210               MSG7:      FNT                          ;SETTIMA LINEA
0210 204C4120                 DEFM  ' LA -T- CORRISPONDE AL DO CENTRALE'
0214 2D542D20
0218 434F5252
021C 4353504F
0220 4E444520
0224 414C2044

   ** Z80 ASSEMBLER SP-2102 PAGE 06  **

0228 4F204345
022C 4E545241
0230 4C45
0232 0D                       DEFB  0DH
0233                          END

        ** Z80 ASSEMBLER SP-2102 PAGE 07  **

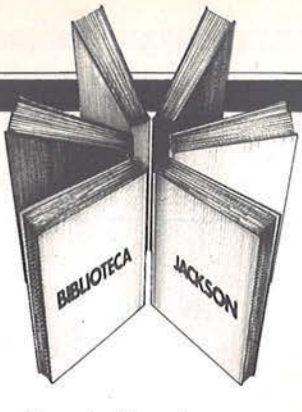
   CMPR    00CA  DISP1   007B  DISP2   0086  DISP3   0091  DISP4   009C
   DISP5   00A7  DISP6   00B2  GETKY   001B  LETNL   0006  MNTR    0000
   MSG     0015  MSG1    0135  MSG2    0155  MSG3    0175  MSG4    019B
   MSG5    01C2  MSG6    01E9  MSG7    0210  MSTA    0044  MSTP    0047
   PRNT    0012  SCTBL   00E3  START   0000  START0  00B8  START1  00BB
```

Listato 2 - *Elenco istruzioni del programma in linguaggio macchina.*

```
:A000=3E 16 CD 12 00 11 35 A1 />.CD...5a
:A008=CD 15 00 CD 06 00 11 55 /CD..CD...
U
:A010=A1 CD 15 00 CD 06 00 11 /aCD..CD..
.
:A018=35 A1 CD 15 00 CD 06 00 /5aCD..CD.
.
:A020=CD 06 00 CD 06 00 11 75 /CD..CD...
75
:A028=A1 CD 15 00 CD 06 00 CD /aCD..CD..
CD
:A030=06 00 11 9B A1 CD 15 00 /...xaCD..
:A038=CD 06 00 CD 15 00 CD 06 /CD..CD..C
D.
:A040=00 CD 15 00 CD 06 00 11 /.CD..CD..
.
:A048=C2 A1 CD 15 00 CD 06 00 /C2aCD..CD
..
:A050=CD 15 00 CD 06 00 CD 15 /CD..CD..C
D.
:A058=00 CD 06 00 CD 06 00 11 /.CD..CD..
.
:A060=E8 A1 CD 15 00 CD 06 00 /E8aCD..CD
..
:A068=CD 06 00 CD 06 00 11 0F /CD..CD...
.
:A070=A2 CD 15 00 3E 07 21 19 /zCD..>.!.
:A078=D9 06 25 77 23 05 C2 7B /D9.%77#.C
2
:A080=A0 21 41 D9 06 25 77 23 /q!AD9.%77
#
:A088=05 C2 86 A0 21 69 D9 06 /.C286q!69
D9.
:A090=25 77 23 05 C2 91 A0 21 /%77#.C291
q!
:A098=91 D9 06 25 77 23 05 C2 /91D9.%77#
.C2
:A0A0=9C A0 21 B9 D9 06 25 77 /dq!ÄD9.%7
7
:A0A8=23 05 C2 A7 A0 21 E1 D9 /#.C2A7q!E
1D9
:A0B0=06 25 77 23 05 C2 B2 A0 /.%77#.C2Ü
q
:A0B8=CD 47 00 CD 1B 00 B7 28 /CDG.CD..o
(
:A0C0=F7 FE 21 CA 00 00 47 21 /F7FE!CA..
G!
:A0C8=E3 A0 7E FE F0 28 E9 23 /E3q7EFEF0
(E9#
:A0D0=B8 28 04 23 23 18 F3 5E /I(.##.F3↑
:A0D8=23 56 ED 53 A1 11 CD 44 /#VEDSa.CD
D
:A0E0=00 18 D8 51 A4 2C 32 00 /..D8Qs,2.
:A0E8=2A 57 A8 27 33 82 25 45 /*Wö'382%E
:A0F0=31 23 34 87 21 52 E3 1F /1#487!RE3
```

Seguito listato 2.

```
:A0F8=54 EE 1D 36 34 1C 59 92 /TEE.64.Ye
:A100=1A 37 1E 19 55 BF 17 49 /.7..UBF.I
:A108=52 16 39 F1 14 4F D4 13 /R.9F1.OD4
.
:A110=30 BC 12 50 98 11 2D C3 /0BC.Ph.-C
3
:A118=10 40 F1 0F 5B F7 0E 5C /.@F1.[F7.
\
:A120=20 0E FC 3D 0D 4C 94 0C / .↓=.L^.
:A128=2E DF 0B 2F 2D 0B 3A 7C /.DF./-.:7
C
:A130=0A 3F EA 09 F0 20 20 20 /.?EA.F0
:A138=20 20 20 20 20 20 2A 2A /       **
:A140=2A 2A 2A 2A 2A 2A 2A 2A /********
:A148=2A 2A 2A 2A 2A 2A 2A 2A /********
:A150=2A 2A 2A 0D 20 20 20 20 /****.
:A158=20 20 20 20 20 20 2A 20 /      *
:A160=4F 52 47 41 4E 4F 20 45 /ORGANO E
:A168=4C 45 54 54 52 4F 4E 49 /LETTRONI
:A170=43 4F 20 2A 0D 20 20 32 /CO *.  2
:A178=20 20 33 20 20 34 20 20 /  3  4
:A180=20 36 20 20 37 20 20 20 / 6  7
:A188=39 20 20 30 20 20 2D 20 /9  0  -
:A190=20 20 5C 20 20 4C 20 20 /  \  L
:A198=20 3A 0D 20 C1 C8 D5 C1 / :. C1C8D
5C1
:A1A0=C8 D5 C1 C8 D5 FD C1 C8 /C8D5C1C8D
5FDC1C8
:A1A8=D5 C1 C8 D5 FD C1 C8 D5 /D5C1C8D5F
DC1C8D5
:A1B0=C1 C8 D5 C1 C8 D5 FD C1 /C1C8D5C1C
8D5FDC1
:A1B8=C8 D5 C1 C8 D5 FD C1 C8 /C8D5C1C8D
5FDC1C8
:A1C0=D5 0D 20 20 FD 20 20 FD /D5.  FD
FD
:A1C8=20 20 FD 20 FD 20 FD 20 /  FD FD F
D
:A1D0=20 FD 20 FD 20 FD 20 20 / FD FD FD
:A1D8=FD 20 20 FD 20 FD 20 FD /FD  FD FD
 FD
:A1E0=20 20 FD 20 FD 20 FD 0D /  FD FD F
D.
:A1E8=20 51 20 57 20 20 45 20 / Q W  E
:A1F0=20 52 20 54 20 59 20 20 / R T Y
:A1F8=55 20 49 20 4F 20 20 50 /U I O  P
:A200=20 20 40 20 5B 20 FC 20 /  @ [ ↓
:A208=20 2E 20 2F 20 3F 0D 20 / . / ?.
:A210=4C 41 20 2D 54 2D 20 43 /LA -T- C
:A218=4F 52 52 49 53 50 4F 4E /ORRISPON
:A220=44 45 20 41 4C 20 44 4F /DE AL DO
:A228=20 43 45 4E 54 52 41 4C / CENTRAL
:A230=45 0D 00 00 00 00 00 00 /E.......
```



Libri firmati JACKSON



Douglas Hergert
IL BASIC NEGLI AFFARI

Il libro insegna a leggere, scrivere e provare programmi BASIC per tipiche applicazioni da ufficio.
Nei primi capitoli viene presentato il linguaggio BASIC e vengono descritti gli elementi fondamentali della programmazione. Ogni gruppo di istruzioni è poi illustrato con esempi di programmi relativi ad applicazioni per l'ufficio e con diversi esercizi; di alcuni di questi sono fornite le soluzioni.
Completa il libro una breve introduzione ad altri tre linguaggi di programmazione: FORTRAN, COBOL e PASCAL, che vengono confrontati con il BASIC.

202 pagine

Codice **402H L. 18.000**

X.T. Bui
LA GESTIONE AZIENDALE CON IL BASIC

Il libro presenta i principali problemi di gestione aziendale, spiegandone i fondamenti teorici e la realizzazione in linguaggio BASIC.
Con i numerosi esempi di applicazioni, ispirati a situazioni reali, l'autore vuole dimostrare che ogni responsabile, che debba prendere decisioni nell'ambito di un'azienda, può accedere direttamente al calcolatore e ottenere elementi di analisi, pianificazione e controllo.

188 pagine

Codice **403H L. 15.000**

Francis Samish
GUIDA ALLA SCELTA DEL PERSONAL COMPUTER

Il segreto per un acquisto intelligente sta nel conoscere bene che cosa si vuole ottenere e, in base a questo, quale hardware e quale software sono necessari.
Questo libro affronta il problema in modo esauriente, fornendo un'ampia panoramica delle macchine che possono essere utili in casa e in ufficio, con il relativo software.
Per chi già possiede un personal, fornisce utili indicazioni su come ottimizzarne l'utilizzo o completare il sistema.

118 pagine

Codice **400P L. 12.000**

La Biblioteca che fa testo

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più L. 3.000 per contributo fisso **spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca ____________

n° ____________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A. | | | | | | | | | | | |

ORDINE MINIMO L. 50.000

GRUPPO EDITORIALE JACKSON

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

nuovidea

PERSONAL SOFTWARE

Dama per ZX81

Un classico per il vostro Sinclair. È richiesta l'espansione da 16 Kbyte

di *Angelo Motta*

Questo programma è stato creato per poter utilizzare lo ZX81 come scacchiera per il gioco della dama fra due persone. Anche se a prima vista può sembrare poco interessante e forse inutile (qualcuno potrebbe dire: perché perdere tanto tempo ad inserire tutte quelle linee quando, a tale scopo, basta una comune scacchiera!), il programma mostra la sua effettiva utilità nelle varie opzioni offerte - più avanti illustrate - che possono essere maggiormente sfruttate da chi possiede una stampante.

Oltre a fare da arbitro al gioco, lo ZX81 vi consente:

- di avere, sulla stampante, una copia della scacchiera in qualsiasi momento della partita, inserendo al posto della mossa di uno dei giocatori la lettera C. Questo può essere utile quando si desidera analizzare - in un secondo tempo - una determinata situazione di gioco;
- di ottenere, sempre sulla printer, inserendo M al posto della mossa, l'elenco di tutte le mosse effettuate, sia durante la gara, sia alla fine della stessa. Questo permette di poter conservare lo sviluppo di una partita ed analizzarla successivamente;
- di registrare la partita ad un determinato punto e completarla in un secondo tempo, inserendo la lettera R al posto della mossa;
- di selezionare la scacchiera, ossia inserire dame e pedine a piacere e

Listato 1. *Il programma Dama.*

```
  10 PRINT TAB 12;"■ ■ ■ ■"
  20 PRINT ,,,,TAB 2;"VOLETE LE
ISTRUZIONI? (S/N)"
  30 GOTO 30+(5 AND INKEY$="S")+
(68 AND INKEY$="N")
  35 CLS
  36 REM
  37 REM ISTRUZIONI
  38 REM
  40 PRINT "ISTRUZIONI PER USO P
ROGRAMMA:"
  45 PRINT ,,"A-LE CASELLE DELLA
 DAMA SONO NU-   MERATE DA 01 A 3
2 PARTENDO DAL   QUADRATO IN BASS
O A DESTRA CO-   ME DAL SEGUENTE
PROSPETTO:"
  50 PRINT ,," 32  31  30  29";
TAB 4;"28  27  26  25";TAB 2;"24
 23  22  21";TAB 4;"20  19  18
17";TAB 2;"16  15  14  13";TAB
4;"12  11  10  09";TAB 2;"08  07
 06  05";TAB 4;"04  03  02  01"
  55 PRINT ,,"B-LA MOSSA VA INSE
RITA INDICANDO   LA CASELLA DI PA
RTENZA SPAZIO    CASELLA DI ARRIV
O"
  57 PRINT ,,"    PREMI [illegible]
N/L CONTINUA    "
  59 IF INKEY$="C" THEN COPY
  61 IF INKEY$<>CHR$ 118 THEN GO
TO 59
  63 CLS
  65 PRINT "  LE CASELLE DALL'1
AL 9 DEVONO      ESSERE INDICATE CO
M 01,02,..09"
  70 PRINT ,,"C-SE SI DEVE PREND
ERE PIU' DI UNA  PEDINA AVVERSARI
A CON UN'UNICA   MOSSA, BISOGNA E
FFETTUARE TAN-   TE MOSSE QUANTE
SONO LE PEDINE   DA PRENDERE, INS
ERENDO AL PO-    STO DELLO SPAZIO
 LA LETTERA P,   SALVO PER L'ULTI
MA PEDINA"
  72 PRINT ,,"D-PER SOFFIARE INS
ERIRE LA CA-     SELLA DELLA PEDI
NA CHE NON VI    E' STATA MANGIAT
A,UNA S AL PO-   STO DELLO SPAZIO
, POI IL NUME-   RO DELLA CASELLA
 DELLA PEDINA    AVVERSARIA DA SO
FFIARE."
  74 PRINT ,,"E-ALTRE OPZIONI OF
FERTE DAL PRO-   GRAMMA:"
  76 PRINT ,,"    PREMI [illegible]
[illegible]    "
  78 IF INKEY$="C" THEN COPY
  80 IF INKEY$<>CHR$ 118 THEN GO
TO 78
  82 CLS
  84 PRINT "  AL POSTO DELLA MOS
SA INSERIRE      LE SEGUENTI LETTER
E PER:"
  86 PRINT ,,"   C=STAMPA LA SCAC
CHIERA SULLA       PRINTER"
  87 PRINT ,,"   M-STAMPA L'ELENC
O DELLE MOSSE      EFFETTUATE SIN
O A QUEL DATO      MOMENTO, SULLA
 PRINTER"
  88 PRINT ,,"   R-SALVA IL PROGR
AMMA CON LA        PARTITA IN COR
SO"
  90 PRINT ,,"   S-PERMETTE DI SE
LEZIONARE LA       SCACCHIERA CON
 PEDINE E DAME     A PIACERE"
  95 PRINT AT 21,0;"     PREMI [illegible]
[illegible] - N/L=INIZIO      "
  96 IF INKEY$="C" THEN COPY
  97 IF INKEY$<>CHR$ 118 THEN GO
TO 96
  98 CLS
  99 FAST
 100 REM
 101 REM VARIABILI INIZIALI
 102 REM
 105 LET PB=12
 110 LET PN=12
 120 DIM A(32)
```

Dama per ZX81

Seguito listato Dama.

```
130 FOR I=1 TO 12
140 LET A(I)=52
150 LET A(I+8)=128
160 LET A(I+20)=180
170 NEXT I
180 DIM B(32)
185 FOR N=0 TO 7
190 FOR I=1 TO 4
200 LET B(I+N*4)=17-N*2
210 NEXT I
220 NEXT N
290 DIM C(32)
295 FOR N=0 TO 24 STEP 8
300 FOR I=1 TO 4
310 LET C(I+N)=34-4*I
320 LET C(I+4+N)=32-4*I
330 NEXT I
340 NEXT N
350 LET M=0
360 LET L$=""
370 DIM M$(32)
400 GOSUB 7000
590 SLOW
697 REM
698 REM CICLO PRINCIPALE
699 REM
700 LET M=M+1
720 LET V=52
730 LET Z=23
735 LET X=5
740 GOTO 2000
750 STOP
800 LET V=180
810 LET Z=151
820 LET X=11
830 GOTO 2000
2000 REM
2001 REM INSERIMENTO E
2002 REM CONTROLLO MOSSA
2003 REM
2004 PRINT AT 18,9;M
2005 PRINT AT X,0;"MUOVE"
2007 PRINT AT 20,0;M$
2010 INPUT A$
2020 IF CODE A$=40 THEN GOTO 500
0
2030 IF CODE A$=50 THEN GOTO 300
0
2040 IF CODE A$=55 THEN GOTO 380
0
2045 IF CODE A$=56 THEN GOTO 510
0
2050 IF LEN A$<>5 THEN GOTO 4000
2060 LET E=VAL A$(1 TO 2)
2070 LET D=VAL A$(4 TO 5)
2080 IF D<1 OR D>32 OR E<1 OR E>
32 THEN GOTO 4000
2085 IF CODE A$(3)=56 THEN GOTO
2500
2100 IF A(D)<>128 OR A(E)<>V AND
 A(E)<>Z THEN GOTO 4000
2110 IF E>D AND A(E)=52 OR D>E A
ND A(E)=180 THEN GOTO 4000
2115 IF ABS (E-D)>5 THEN GOTO 35
00
2120 IF ABS (C(E)-C(D))<>2 THEN
GOTO 4000
2130 PRINT AT X,0;A$
2140 LET A(D)=A(E)
2150 IF A(E)=V AND D<=4 OR D>=29
 THEN LET A(D)=Z
2160 LET A(E)=128
2170 PRINT AT B(D),C(D);CHR$ (A(
D));AT B(E),C(E);CHR$ (A(E))
2190 LET L$=L$+A$
2195 IF PN=0 OR PB=0 THEN GOTO 4
500
2200 IF CODE A$(3)=53 OR CODE A$
(3)=56 THEN GOTO 2000
2300 GOTO 700+(100 AND V=52)
2500 REM
2501 REM SOFFIO DELLA PEDINA
2502 REM
2510 IF A(E)<>V AND A(D)=V OR A(
D)=128 THEN GOTO 4000
2520 IF ABS (E-D)<>5 AND ABS (E-
```

quindi iniziare la gara da quella posizione di gioco. Opzione che si rivela utile, quando si vuol risolvere i problemi di dama, pubblicati su giornali e riviste.

Per quanto riguarda il gioco, è molto importante tenere ben presente come va inserita la mossa: "casella di partenza spazio casella di arrivo", ricordandosi che per le prime nove caselle, tale numero dovrà essere preceduto da uno zero (esempio 01, 02, ... 09).
Se nel corso della gara dovrete prendere più di una pedina avversaria con un'unica mossa, dovrete effettuare tante mosse quante sono le pedine da prendere, inserendo al posto dello spazio la lettera P, ad eccezione dell'ultima mossa. Esempio: se siete in casella 16 e dovete prendere le pedine avversarie nelle caselle 20 - 27 - 26, dovrete inserire tre mosse in successione come le seguenti: 1) 16P23 - 2) 23P30 - 3) 30spazio21.
Si è ricorso a questa procedura, in quanto avere la presa di più pedine con un'unica mossa, avrebbe comportato un notevole allungamento del programma a seguito dei necessari controlli e, tenuto conto che durante una partita non sono molte le mosse di questo tipo, si è optato per la soluzione sopra illustrata evitando, perciò, di appesantire ulteriormente il programma.

Le istruzioni per l'uso sono contenute nel programma stesso. È stata prevista la possibilità di copiarle sulla stampante in modo di averle sempre a disposizione nel corso della gara. Questa soluzione risulta comoda per non dimenticarsi della numerazione delle caselle e delle varie opzioni possibili. Per quanto riguarda l'illustrazione del programma si vedano le Rem inserite, che dividono i vari blocchi di cui è composto

Seguito listato Dama.

```
D)<>4 AND ABS (E-D)<>3 THEN GOTO
 4000
2530 LET A(D)=128
2540 PRINT AT B(D),C(D);CHR$ A(D
)
2550 GOSUB 4100
2560 GOTO 2190
3000 REM
3001 REM STAMPA MOSSE
3002 REM
3010 GOSUB 3030
3020 GOTO 2010
3030 IF L$<>"" THEN GOTO 3060
3040 LPRINT "NON E' STATA EFFETT
UATA ALCUNA  MOSSA - DOVETE PRIM
A GIOCARE"
3050 RETURN
3060 LPRINT "ELENCO DELLE MOSSE
EFFETTUATE:"
3070 LPRINT
3080 LPRINT "BIANCO","NERO"
3085 LPRINT
3090 FOR I=1 TO LEN L$ STEP 5
3100 LPRINT L$(I TO I+4),
3110 IF CODE L$(I+2)<>0 THEN LPR
INT ,
3120 NEXT I
3125 LPRINT
3130 RETURN
3500 REM
3501 REM PRESA DI PEDINA
3502 REM
3510 LET F=INT ((E+D)/2)
3520 IF C(D)-INT (C(D)/4)*4=0 TH
EN LET F=F+1
3530 IF V=180 THEN GOTO 3600
3540 IF A(E)=52 AND A(F)=151 THE
N GOTO 4000
3550 IF A(F)<>151 AND A(F)<>180
THEN GOTO 4000
3560 GOTO 3700
3500 IF A(F)<>52 AND A(F)<>23 TH
EN GOTO 4000
3610 IF A(E)=180 AND A(F)=23 THE
N GOTO 4000
3700 LET A(F)=128
3710 PRINT AT B(F),C(F);CHR$ A(F
)
3740 GOSUB 4100
3780 GOTO 2130
3800 REM
3801 REM REGISTRAZ.PARTITA
3802 REM
3810 PRINT AT 20,0;"VUOI REGISTR
ARE LA PARTITA?(S/N)"
3820 IF INKEY$="S" THEN GOTO 386
0
3830 IF INKEY$<>"N" THEN GOTO 38
20
3840 PRINT AT 20,0;M$
3850 GOTO 2000
3860 PRINT AT 20,0;"FAI PARTIRE
IL REGISTRATORE E   POI PREMI NE
WLINE"
3870 INPUT Z$
3880 FOR I=1 TO 50
3890 NEXT I
3900 SAVE "DAMA"
3910 LET M=M-1
3920 GOSUB 7000
3930 GOTO 700+(100 AND V=180)
4000 REM
4001 REM MOSSA ERRATA
```

Dama per ZX81

Seguito listato Dama.

```
4002 REM
4010 PRINT AT X,0;"MOSSA ERRATA"
4020 FOR I=1 TO 70
4030 NEXT I
4040 PRINT AT X,0;"                "
4050 GOTO 2000
4100 REM
4101 REM AGG.TO PEDINE
4102 REM
4110 IF V=180 THEN LET PB=PB-1
4120 IF V=52 THEN LET PN=PN-1
4130 RETURN
4500 REM
4501 REM FINE PARTITA
4502 REM
4510 PRINT AT 20,0;"IL "+("NERO"
 AND PB=0)+("BIANCO" AND PN=0);"
 VINCE"
4520 PRINT "[illegible]
[illegible]"
4530 IF INKEY$="S" THEN GOSUB 30
60
4540 IF INKEY$<>"I" THEN GOTO 45
30
4550 GOTO 95
5000 REM
5001 REM COPY SU PRINTER
5002 REM
5005 COPY
5010 GOTO 2010
5100 REM
5101 REM SELEZIONE SCACCHIERA
5102 REM
5110 LET PN=0
5120 LET PB=0
5125 LET L$=""
5130 FOR I=1 TO 32
5140 PRINT AT B(I),C(I);CHR$ 148
5145 PRINT AT 20,0;"■=PEDINA ■=
DAMA ■■■=AVANZA"
5150 INPUT R$
5160 IF CODE R$<>53 AND CODE R$<
>41 THEN GOTO 5280
5170 PRINT AT 20,0;"■ = NERO ■
 = BIANCO
5180 INPUT S$
5190 IF CODE S$<>51 AND CODE S$<
>39 THEN GOTO 5180
5200 LET A(I)=52+(128 AND CODE S
$=51)-(29 AND CODE R$=41)
5210 LET PB=PB+(CODE S$=39)
5220 LET PN=PN+(CODE S$=51)
5230 PRINT AT B(I),C(I);CHR$ A(I
)
5240 NEXT I
5250 PRINT AT 20,0;"CHI DEVE MUO
VERE?  (B / N)
5260 LET M=1
5270 GOTO 5270-(4570 AND INKEY$=
"B")-(4470 AND INKEY$="N")
5280 LET A(I)=128
5290 GOTO 5230
5998 STOP
5999 SAVE "DAM■"
6000 GOTO 1
7000 REM
7001 REM STAMPA SCACCHIERA
7002 REM
7010 FOR N=1 TO 13 STEP 4
7020 FOR I=1 TO 13 STEP 4
7030 PRINT AT 2+N,15+I;"■■ ■";TA
B 15+I;"■■■■";TAB 15+I;"  ■■■";TA
B 15+I;"■■■ "
7040 NEXT I
7050 NEXT N
7060 FOR I=1 TO 13 STEP 4
7070 PRINT AT 2,15+I;"■■■■"
7080 PRINT AT 18,14+I;"■■■■■"
7090 PRINT AT 2+I,15;"■";TAB 15;
"■";TAB 15;"■";TAB 15;"■"
7100 PRINT AT 1+I,31;"■";TAB 31;
"■";TAB 31;"■";TAB 31;"■"
7110 NEXT I
7120 PRINT AT 18,31;"■";AT 2,15;
"■"
7130 FOR I=1 TO 32
7140 PRINT AT B(I),C(I);CHR$ A(I
)
7150 NEXT I
7160 PRINT AT 0,12;"■ ■ ■ ■"
7170 PRINT AT 3,0;"O - BIANCO";A
T 9,0;"■ - NERO"
7180 PRINT AT 18,0;"MOSSA N. "
7190 RETURN
```



Figura 1. *La scacchiera, come si presenta all'inizio di una partita.*

VARIABILI UTILIZZATE

PB	Numero delle pedine bianche.
PN	Numero delle pedine nere.
A(32)	Matrice contenente il valore delle caselle della scacchiera: 52 = pedina bianca, 23 = dama bianca, 180 = pedina nera, 151 = dama nera, 128 = casella vuota.
B(32)	Matrice contenente le coordinate orizzontali delle caselle della scacchiera.
C(32)	Matrice contenente le coordinate verticali delle caselle della scacchiera.
M	Numero della mossa.
L$	Stringa contenente le mosse effettuate da entrambi i giocatori.
M$	Maschera per la cancellazione del video.
V	Valore della pedina nel corso della gara.
Z	Valore della dama nel corso della gara.
X	Punto di stampa a video della mossa del giocatore.
A$	Mossa corrente.
D-E-F	Valore delle caselle durante la mossa effettuata.
R$	Valore della casella in fase di selezione della scacchiera.
S$	Colore della pedina o dama in fase di selezione della scacchiera.

Il Castello

Un adventure per lo Spectrum 48 Kbyte

di *Luciano Lotti*

Per gli sfortunati che ancora non sapessero cos'è un adventure eccone una descrizione! Si tratta di giochi che proiettano idealmente il giocatore in un mondo fantastico, ricco di cose strane, pieno di pericoli e di enigmi che mettono a dura prova la fantasia.
Lo scopo è di riuscire a "portare a casa la pellaccia" scoprendo il modo di attraversare le varie prove che il computer ci presenta.
Non aspettatevi però di veder comparire immagini in movimento ed effetti sonori: la descrizione degli ambienti in cui ci si muove è tutta dialogata (alcuni programmi presentano anche delle schermate grafiche).
Ciò è dovuto in parte al fatto che le capacità grafiche dei primi elaboratori su cui sono stati implementati questi programmi erano scarse e che esse comunque richiedono vaste aree di memoria, ma in parte è voluto: la nostra fantasia, infatti, può riuscire a trasformare poche decine di parole in un'immagine ben più reale di qualunque disegno (per questo molti considerano le avventure grafiche un regresso).

I soggetti degli adventure spaziano dai castelli inglesi con fantasmi e passaggi segreti, alle piramidi con mummie e teschi, al mondo delle fate con orchi, folletti, giganti e simili.
Non si deve però credere che per questo siano giochi per bambini: tutt'altro, basta solamente dire che anche le avventure più semplici richiedono ore e ore per essere risolte.
Per passare attraverso un portone è

Listato 1. *Il programma Castello.*

```
 100 GO SUB 6000: REM iniz
 110 GO SUB 6800: REM present
 120 IF luogo=99 THEN GO SUB 690
0: GO TO 9990
 130 GO SUB 6500: REM █descriz█
 135 CLS
 140 PRINT l$
 150 GO SUB 7000: REM █ oggetti█
 200 GO SUB 6600: REM █accetta m
os█
 500 IF luogo<>28 OR NOT gf THEN
 GO TO 800
 510 REM █ GOBBO █
 515 IF v$<>"uccidi" THEN GO TO
700
 520 IF par<2 THEN INPUT "Uccidi
 chi?";b$: IF b$="" THEN GO TO 5
20
 521 IF par<2 THEN LET par=2
 525 IF b$<>"il gobbo" THEN GO T
O 700
 530 IF par<3 THEN INPUT "Con co
sa devo ucciderlo ?";c$: IF c$="
" THEN GO TO 530
 531 LET par=3
 535 IF c$<>"con l'ascia" THEN G
O TO 700
 540 IF c$="l'ascia" AND l(2) TH
EN PRINT '"Non ho l'ascia !": GO
 TO 700
 545 GO TO 800
 700 PRINT '"Il gobbo si avventa
 su di te e  con le sole mani ti
 spezza la   colonna vertebrale"
: GO TO 9990
 800 IF luogo<>4 OR NOT vf THEN
GO TO 1000
 810 REM █ VAMPIRO █
 815 IF v$<>"mostra" THEN GO TO
950
 820 IF par<2 THEN INPUT "Mostra
 a chi ?";b$: IF b$="" THEN GO T
O 820
 825 IF b$<>"al vampiro" THEN GO
 TO 950
 830 IF par<3 THEN INPUT '"Cosa
devo mostrare al vampiro ?";c$:
IF c$="" THEN GO TO 830
 835 IF c$<>"la croce" AND c$<>"
la croce d'oro" THEN GO TO 950
 840 IF l(6) THEN PRINT "Non ho
la croce d'oro": GO TO 950
 850 GO TO 1000
 950 PRINT '"Il vampiro ti affer
ra e ti az-  zanna il collo.Ti t
rasformi in  un vampiro e'sei co
ndannato a   vagare eternamente
nel castello": GO TO 9990
1000 IF v$<>"n" THEN GO TO 1060
1005 REM █ NORD█
1010 IF s$(luogo)(2 TO 3)="00" T
HEN PRINT '"Non si puo'andare a
nord": GO TO 200
1015 IF s$(luogo,1)=" " THEN GO
TO 1050
1020 IF s$(luogo,1)<>"#" THEN PR
INT '"C'e'la porta che blocca il
 pas- saggio": GO TO 200
1050 LET luogo=VAL s$(luogo)(2 T
O 3): GO TO 120
1060 IF v$<>"s" THEN GO TO 1100
1065 REM █ sud █
1070 IF s$(luogo)(5 TO 6)="00" T
HEN PRINT '"Non si puo andare a
sud": GO TO 200
1075 IF s$(luogo,4)=" " THEN GO
TO 1090
1080 IF s$(luogo,4)<>"#" THEN PR
INT '"C'e'la porta che blocca il
 pas- saggio": GO TO 200
```

```
1090 LET luogo=VAL s$(luogo)(5 T
O 6): GO TO 120
1100 IF v$<>"o" THEN GO TO 1150
1110 REM ■ ovest ■
1115 IF s$(luogo)(8 TO 9)="00" T
HEN PRINT '"Non si puo'andare a
ovest": GO TO 200
1120 IF s$(luogo,7)=" " THEN GO
TO 1130
1125 IF s$(luogo,7)<>"#" THEN PR
INT '"C'e' la porta che blocca i
l pas-saggio": GO TO 200
1130 LET luogo=VAL s$(luogo)(8 T
O 9): GO TO 120
1150 IF v$<>"e" THEN GO TO 1200
1155 REM ■ est■
1160 IF s$(luogo)(11 TO 12)="00"
 THEN PRINT '"Non si puo'andare
ad est": GO TO 200
1165 IF s$(luogo,10)=" " THEN GO
 TO 1175
1170 IF s$(luogo,10)<>"#" THEN P
RINT '"C'e'la porta che blocca i
l pas- saggio": GO TO 200
1175 LET luogo=VAL s$(luogo)(11
TO 12): GO TO 120
1200 IF v$<>"sali" THEN GO TO 12
50
1205 REM ■sali■
1220 IF luogo<>16 AND luogo<>24
THEN PRINT '"Non e'possibile sal
ire": GO TO 200
1230 LET luogo=VAL s$(luogo)(14
TO 15): GO TO 120
1250 IF v$<>"scendi" THEN GO TO
1300
1255 REM ■ scendi■
1260 IF luogo<>16 AND luogo<>24
THEN PRINT '"Non e'possibile sce
ndere": GO TO 200
1270 LET luogo=VAL s$(luogo)(17
TO 18): GO TO 120
1300 IF v$<>"apri" THEN GO TO 14
00
1305 REM ■ apri ■
1310 IF b$="la bottiglia" AND NO
T (luogo=l(12) OR l(12)=0) THEN
PRINT "Stai forse sognando, non
vedo  alcuna bottiglia!": GO TO
 200
1311 IF b$="la bottiglia" THEN P
RINT "E'vuota e senza tappo": GO
 TO 200
1313 IF par<2 THEN INPUT " APRO
COSA ?";b$: IF b$="" OR b$=" " T
HEN GO TO 1313
1314 IF LEN b$<11 THEN GO TO 131
9
1315 IF b$( TO 9)="la porta " TH
EN LET c$=b$(10 TO ): LET par=3
1319 IF b$<>"la porta" THEN PRIN
T "Non capisco come si possa apr
ire";b$: GO TO 200
1320 LET r$=" CLUNK!!! "
1325 IF par<3 THEN INPUT "In qua
le direzione ?";c$
1326 IF c$="" OR c$=" " THEN GO
TO 1325
1330 LET dir=0+(c$="a nord" OR c
$="n")+(4 AND (c$="a sud" OR c$=
"s"))+(7 AND (c$="a ovest" OR c$
="o"))+(10 AND (c$="a est" OR c$
="e"))
1340 IF dir=0 THEN PRINT '"Non h
o capito quale porta": LET par=2
: GO TO 1325
1350 IF s$(luogo,dir)=" " THEN P
RINT "In questa direzione non c'
e'al- cuna porta": GO TO 200
1353 IF s$(luogo,dir)="#" THEN P
RINT '"E'gia stata aperta": GO T
O 200
1355 LET k$=s$(luogo,dir)
```

spesso necessario ricercare le chiavi poste, statene certi, in luoghi impensati custoditi da esseri strani che possono essere eliminati con l'ausilio di oggetti situati in posti ancora più stravaganti.

Naturalmente non mancano per completare l'opera vampiri, zombie, trabocchetti, pozioni velenose che provocano prematuri decessi e la necessità di ricominciare dall'inizio.

In questi mondi il giocatore non può però agire direttamente, ma deve ordinare al computer cosa fare per lui. Si comanda un automa che per noi vede, agisce, vaga per i paesaggi e (sigh!) subisce.

Gli ordini al calcolatore vengono solitamente impartiti mediante semplici frasi, in linguaggio naturale, costituite da un verbo o da un verbo e un nome come: "prendi il diamante", "sali", "uccidi il folletto", "nuota", ecc.

I verbi ammessi e i nomi riconosciuti non sono molti, ma vi assicuro che ciò non toglie nulla al divertimento, anzi il più delle volte costituisce una facilitazione nelle situazioni particolarmente arrovellate.

Alcuni programmi commerciali, quali ad esempio "The Hobbit", permettono di usare per le comunicazioni preposizioni ed avverbi che rendono il dialogo molto più naturale.

Questo programma in particolare visualizza anche numerosi ambienti, esistenti anche nel libro da cui è stato tratto e di cui riproduce fedelmente la trama, e possiede due accorgimenti particolari: l'Animtalk e l'Animaction.

L'Animtalk permette di comunicare con gli altri personaggi dell'avventura rispondendo alle loro domande e dicendogli cosa si vuole che facciano.

L'Animaction invece dota i vari personaggi di un "anima" propria per

cui ad ogni nuova partita si possono comportare in modo leggermente diverso; se le richieste poste loro non sono cortesi, o il loro umore è un po' "nero", ci si può facilmente aspettare un secco "No!" come risposta.

Strategie risolutive

Ogni adventure ha una propria strategia risolutiva che va scoperta per tentativi ed errori.
Sbagli in situazioni particolarmente delicate causano, come già accennato, una fine prematura che implica la ripetizione dall'inizio delle mosse che hanno portato a quel punto e la variazione dell'ultima azione che ha causato la penosa conclusione.
Un valido aiuto per la soluzione è costituito da mappe che indichino le relazioni tra i vari ambienti ed eventualmente le localizzazioni dei vari oggetti e pericoli.
Non vi resta perciò che armarvi di carta e matita e, man mano che procedete, aggiungere le nuove scoperte.

Il castello

Il Castello è un adventure che si

VARIABILI NUMERICHE

altra	Usato per levare il marcatore di porta chiusa.
apr	Flag che indica se si ha la chiave giusta.
dir	Contiene un codice corrispondente alla direzione ed è usato nell'apertura delle porte.
fla	Usato per indicare se lo spazio è tra il nome e l'articolo o dopo il nome.
g	Usato in vari cicli.
gf	Flag che indica se il gobbo è vivo.
luogo	Contiene il numero corrispondente al luogo in cui ci si trova.
ogpr	Numero di oggetti portati.
pas	Numero di parti di cui è composta la frase in input.
pun	Puntatore ultimo carattere analizzato della stringa.
tro	Flag che indica se l'oggetto è presente.
uno	Flag che indica se è già stato stampato il messaggio.
vf	Flag che indica se il vampiro è vivo.

VARIABILI ALFANUMERICHE

b$	Contiene il secondo elemento della frase in input.
c$	Contiene il terzo elemento della frase in input.
g$	Contiene il simbolo della condizione necessaria per lo spostamento in un altro locale.
i$	Usato per l'input.
k$	Indica il tipo di chiave necessaria per aprire la porta.
l$	Contiene la descrizione del luogo.
v$	Contiene il verbo della frase in input.
z$	Usata per rendere uniforme la lunghezza di b$ nelle frasi "prendi xxx", "posa xxx".

VETTORI

o$	Contiene il nome degli oggetti.
s$	Tabella per i cambiamenti di stanza.
l	Contiene i luoghi degli oggetti.

LE PRINCIPALI ROUTINE

100	Chiama inizializzazione.
110	Chiama presentazione.
120	Controlla se hai terminato l'avventura.
130-150	Stampa la descrizione del luogo.
200	Chiama accetta-mossa.
500-700	Procedure per il gobbo.
800-950	Procedure relative al vampiro.
1000-1050	Movimento a nord.
1060-1100	Movimento a sud.
1110-1130	Movimento a ovest.
1150-1175	Movimento a est.
1200-1230	Sali.
1250-1270	Scendi.
1300-1390	Apri.
1400-1460	Prendi.
1500-1560	Posa.
1600-1645	Inventario.
1650-1700	Mostra.
1750-1760	Inizio (per ricominciare).
1765-1725	Fine (per smettere).
1800-1825	Premi.
1850-1870	Aggrappati.
1900	Guarda.
1910-1940	Uccidi.
2000-2025	Mangia.
5598	Il verbo non è stato riconosciuto.
6000-6300	Inizializzazione.
6500-6530	Descrizioni.
6600-6740	Accetta frase.
6800-6899	Presentazione.
7000-7060	Stampa gli oggetti presenti.
9990-9994	Procedure di fine.

n	
s	
e	
o	
sali	
scendi	
apri	(cosa)
prendi	(cosa)
posa	(cosa)
inventario	
mostra	(a chi) (che cosa)
inizio	
fine	
premi	(cosa)
aggrappati	(a cosa)
uccidi	(chi) (con cosa)
mangia	(cosa)
guarda	

Tabella 1. *Questi sono i verbi e i comandi permessi nelle frasi di input. Tra le parentesi sono indicate le eventuali specifiche da dare ad ogni verbo.*

svolge in un maniero abbandonato. Lo scopo è di uscire dalla porta d'oro che bisogna aprire con la chiave d'oro.

I verbi ammessi sono indicati nella tabella 1.

Le eventuali parentesi che li seguono indicano le informazioni complementari da digitare e il loro ordine: "uccidi il mostro con la lancia" è una frase corretta, "uccidi con la spada il mostro" non lo è.

Per prendere un oggetto occorre scrivere per esteso il suo nome e ciò resta valido se lo si vuole posare.

Il numero massimo di oggetti trasportabili è due.

Il comando "inventario" permette di conoscere cosa si stà portando.

Per muoversi da una stanza all'altra si deve digitare l'iniziale del punto cardinale corrispondente alla direzione verso la quale ci si vuole muovere.

Per aprire le porte usate la formula fissa "apri la porta" seguita o dall'iniziale di un punto cardinale o dal suo nome per esteso ("a nord", "a est", ecc.).

Onde evitare errori di riconoscimento dei comandi seguite attentamente le seguenti norme: non usate le maiuscole, separate le parole con un solo spazio e, soprattutto, non ponete spazi dopo l'ultima lettera dell'ultima parola.

Fate anche attenzione a digitare i Data perché errori in queste istruzioni possono alterare tutta la logica degli spostamenti fra le stanze.

Importante: se digitate voi il programma avete praticamente risolto il 90% dell'avventura.

Vi consiglio pertanto di far battere il listato 1 ad un amico (deve proprio esservi amico, se accetta), o di suddividere questo compito tra più persone. Ultimo consiglio per i principianti: attenzione, non è detto che tutti gli oggetti siano effettivamente utili! ■

Seguito listato Castello.

```
1360 LET apr=0+(k$="r" AND NOT l
(7))+(k$="n" AND NOT l(4))+(k$="
o" AND NOT l(5))+(k$="f" AND NOT
 l(3))
1365 IF apr=0 THEN PRINT '"Non h
o la chiave adatta": GO TO 200
1370 LET s$(luogo,dir)="#": LET
altra=VAL s$(luogo)(dir+1 TO dir
+2)
1372 IF altra=99 THEN GO TO 1390
1375 LET dir=(1 AND dir=4)+(4 AN
D dir=1)+(7 AND dir=10)+(10 AND
dir=7)
1380 LET s$(altra,dir)="#"
1390 PRINT r$''"La porta ";c$;"
e'aperta": GO TO 200
1400 IF v$<>"prendi" THEN GO TO
1500
1404 IF par<2 THEN INPUT "Prendi
 cosa ?";b$: LET c$="": IF b$=""
 THEN GO TO 1404
1405 REM ■ prendi ■
1406 IF b$="la moneta" OR b$="la
 chiave" OR b$="la croce" OR b$=
"la sfera" THEN LET b$=b$+" "+c$
1410 IF ogpr=2 THEN PRINT "Non p
uoi portare piu'di due og- getti
.": GO TO 200
1415 LET tro=0: LET b$=b$+z$( TO
 22-LEN b$)
1420 FOR g=1 TO 13
1425 IF b$<>o$(g) THEN GO TO 145
0
1430 LET tro=1: IF l(g)=0 THEN P
RINT '"Guarda che ho'gia'";b$: L
ET g=13: GO TO 1450
1435 IF l(g)<>luogo THEN PRINT "
Qui non c'e'";b$: LET g=13: GO T
O 1450
1440 PRINT '" OK !": LET l(g)=0:
 LET g=13: LET ogpr=ogpr+1
1450 NEXT g
1455 IF NOT tro THEN PRINT b$,"n
on e'un oggetto che posso pren-d
ere"
1460 GO TO 200
1500 IF v$<>"posa" THEN GO TO 16
00
1505 REM ■ posa ■
1506 IF b$="la moneta" OR b$="la
 chiave" OR b$="la croce" OR b$=
"la sfera" THEN LET b$=b$+" "+c$
1510 IF ogpr=0 THEN PRINT '"Non
ho oggetti con me": GO TO 200
1515 LET tro=0: LET b$=b$+z$( TO
 22-LEN b$)
1520 FOR g=1 TO 13
1525 IF b$<>o$(g) THEN GO TO 155
0
1530 LET tro=1: IF l(g)<>0 THEN
PRINT '"Non ho in mano ";b$: LET
 g=13: GO TO 1550
1540 PRINT '" OK !": LET l(g)=lu
ogo: LET g=13: LET ogpr=ogpr-1
1550 NEXT g
1555 IF NOT tro THEN PRINT b$,"n
on e'un oggetto che posso por- t
are"
1560 GO TO 200
1600 IF v$<>"inventario" THEN GO
 TO 1650
1605 REM ■ invent.■
1610 IF ogpr=0 THEN PRINT '"Non
sto'portando niente": GO TO 200
1620 PRINT '"Ho in mano:"
1625 FOR g=1 TO 13
1630 IF NOT l(g) THEN PRINT o$(g
)
1640 NEXT g
1645 GO TO 200
1650 IF v$<>"mostra" THEN GO TO
1750
1655 REM ■ mostra ■
1660 IF par<2 THEN INPUT "Mostra
 a chi ?";b$: IF b$="" THEN GO T
O 1660
1665 IF par<3 THEN INPUT "Cosa d
evo mostrare ?";c$: IF c$="" THE
N GO TO 1665
1670 IF b$<>"al vampiro" THEN GO
 TO 1700
1672 IF vf=0 THEN PRINT '"Il vam
piro e'gia'stato dissolto": GO T
O 200
1675 IF luogo<>4 THEN PRINT "Non
 ci sono vampiri ! Hai forse bev
uto un goccetto di troppo ?": GO
 TO 200
1680 IF c$<>"la croce d'oro" THE
N GO TO 200
1690 LET vf=0: PRINT '"Il vampir
o alla vista della cro-ce si dis
solve in una nuvola  dipolvere":
 PRINT ''" PLUFF !!": GO TO 200
1700 PRINT "Non serve a niente m
ostrare ";b$;" ";c$: GO TO 200
1750 IF v$<>"inizio" THEN GO TO
1765
1755 REM ■ inizio ■
1760 RUN
1765 IF v$<>"fine" THEN GO TO 18
00
1770 REM ■ fine ■
1775 STOP
1800 IF v$<>"premi" THEN GO TO 1
850
1805 REM ■ premi ■
1810 IF (b$<>"il bottone rosso"
AND b$<>"il bottone") THEN PRINT
 "L'unica cosa che potrebbe esse
reinteressante da premere sarebb
e un bottone rosso"
1815 IF luogo<>26 THEN PRINT "Ma
 non vedo il bottone!": GO TO 20
0
1820 IF (b$="il bottone rosso" O
R b$="il bottone") AND luogo=26
THEN PRINT '"Gli scaffali della
cantina stan-no cadendo,proprio
addos...": PRINT '"       CRASH !
!  CRINKLE !!": GO TO 9990
1825 GO TO 200
1850 IF v$<>"aggrappati" THEN GO
 TO 1900
1855 REM ■ aggrap ■
1860 IF par<2 THEN INPUT "Aggrap
pati a che ?";b$: IF b$="" THEN
GO TO 1860
1865 IF b$<>"alla corda" THEN PR
INT "Non ci si puo'aggrappare":
GO TO 200
1867 IF luogo<>9 THEN PRINT '"St
ai prendendomi per fesso !    No
n c'e'nessuna corda": GO TO 200
1870 PRINT '"Il soffitto non sor
regge il tuo peso e crolla seppe
llendoti ": GO TO 9990
1900 IF v$="guarda" THEN GO TO 1
20
1910 IF v$<>"uccidi" THEN GO TO
2000
1915 REM ■ uccidi ■
1920 IF par<2 THEN INPUT "Uccidi
```

Seguito listato Castello.

```
 chi ?";b$: IF b$="" THEN GO TO
1920
1925 IF b$<>"il vampiro" AND b$<
>"il gobbo" THEN PRINT "Stai sch
erzando vero ?": GO TO 200
1930 IF b$="il vampiro" AND (luo
go<>4 OR NOT vf) THEN PRINT "Non
 ci sono vampiri qui ! Penso che
 il tuo equilibrio mentale   sti
a vacillando.": GO TO 200
1935 IF b$="il gobbo" AND (luogo
<>28 OR NOT gf) THEN PRINT '"Hai
 le traveggole ? Non ci sono gob
bi.": GO TO 200
1940 IF c$="con l'ascia" THEN LE
T gf=0: PRINT '"Il gobbo stramaz
za al suolo e    finisce su una b
otola che si     apre e dopo aver
lo fatto preci- pitare si richiu
de inesorabil-   mente dietro di
lui!": GO TO 200
2000 IF v$<>"mangia" THEN GO TO
2050
2005 REM ■ Mangia ■
2010 IF par<2 THEN INPUT "Mangia
 cosa ?";b$: IF b$="" THEN GO TO
 2010
2015 IF b$<>"il biscotto" THEN P
RINT '"Non dire idiozie": GO TO
200
2020 IF l(11) THEN PRINT '"Non s
to'portando il biscotto": GO TO
200
2025 PRINT "Il biscotto e'avvele
nato": GO TO 9990
5998 PRINT ,"■";v$;"■"'"Non e'un
 verbo utilizzabile": GO TO 200
5999 STOP
6000 REM ■inizializ.■
6010 DIM o$(13,22): DIM l(13)
6020 RESTORE 6020: FOR g=1 TO 13
: READ o$(g),l(g): NEXT g
6030 DATA "il pugnale",17,"l'asc
ia",6,"la chiave di ferro",30,"l
a chiave nera",27,"la chiave d'o
ro",11,"la croce d'oro",20,"la c
hiave rossa",14,"la sfera di cri
stallo",18,"il diamante",23,"l'a
nello",7
6035 DATA "il biscotto",15,"la b
ottiglia",26,"la moneta d'argent
o",13
6040 DIM s$(30,18)
6050 RESTORE 6050: FOR g=1 TO 30
: READ s$(g): NEXT g
6060 DATA " 02 14r15o99 00 00","
r03 01 00 00 00 00"," 04r02 00 0
0 00 00"," 00 03 05 00 00 00","
00 00r06 04 00 00"
6065 DATA " 00 07 00r05 00 00","
06 08 00 00 00 00"," 07 09 00 0
0 00 00"," 08 10 00 00 00 00","
09f11 00 00 00 00"
6070 DATA "f10 00 00f12 00 00","
 00 00f11r13 00 00","r14 00r12 0
0 00 00"," 01r13 00 00 00 00","
16 00 00r01 00 00"
6075 DATA " 00 00 00 00 15 17","
 24 16n18 00 00 00"," 00r19r23n1
7 00 00","r18r20r22 00 00 00","r
19 00r21 00 00"
6080 DATA "r22 00 00r20 00 00","
r23r21 00r19 00 00"," 00 22 00r1
8 00 00"," 00 00 00 00 17 25","
00 24r26 27 00 00"
6085 DATA " 00 00 00r25 00 00","
 00 00 25 28 00 00"," 00 29 27 0
0 00 00"," 28r30 00 00 00 00","r
29 00 00 00 00 00"
6100 LET luogo=1
6110 LET vf=1: LET gf=1: LET ogp
r=0
6115 LET z$="
6300 RETURN
6500 GO TO 6500+luogo
6501 LET l$="Sei in una stanza d
i forma qua- drata con una porta
 d'oro a est,una porta rossa a o
vest e due    passaggi nelle altr
e due dire-   zioni. Non ci sono
mobili ne fi-nestre.": RETURN
6502 LET l$="Sei in un corridoio
 con i muri   colorati di azzurro
.Sul soffit-  to c'e un affresco
rovinato in   molti punti. C'e'un
a porta ros-  sa a nord ed un pas
saggio a sud.": RETURN
6503 LET l$="Sei in un corridoio
 che ha due   feritoie:da quella
rivolta a estsi puo'osservare il
 parco ed il sentiero che conduc
e fuori dal   parco che circonda
il castello. Quella a ovest e'tr
oppo in alto perche'riesca a ved
ere.C'e'una   porta rossa a sud e
 si puo an-   dare a nord.Sopra l
a porta c'e'  l'immagine di un pi
pistrello.": RETURN
6504 LET l$="Sei in una stanza s
carsamente    illuminata dalla lu
ce provenien-te dai passaggi a o
vest e a sud."+(" C'e'un vampiro
 che si avvicina minacciosamente
" AND vf): RETURN
6505 LET l$="Sei in un corridoio
 illuminato   da un lucernario su
l soffitto.   Sopra la porta ross
a ad ovest    c'e'un quadro che r
appresenta unlupo.Sul muro ad es
t, sopra un   passaggio, c'e'un o
mbra come se qualcuno avesse lev
ato una cor-  nice.": RETURN
6506 LET l$="Sei in un'ampia sta
nza ammobi-   liata.Piccole ragna
tele sono at-taccate ovunque.C'e
'una porta a est di color rosso
ed un passag-gio a sud.": RETURN

6507 LET l$="Sei in un corridoio
 con passaggia sud e nord.Da una
 feritoia si puo'vedere il lago
ad ovest del castello.": RETURN
6508 LET l$="Sei in un corridoio
 con passaggia sud e nord.Da una
 feritoia si puo'vedere il lago
ad ovest del castello.": RETURN
6509 LET l$="Sei in uno stretto
corridoio e   dal centro del soff
itto pende    una grossa corda.Le
 uscite sono a nord e sud.": RET
URN
6510 LET l$="Sei in un corridoio
 senza fine-  stre. Sul soffitto
probabilmen-  te era dipinta una
scena di cac-cia . A nord c'e'un
 passaggio e a sud c'e'una gross
a porta di    ferro.": RETURN
6511 LET l$="Sei in una stanza d
i forma otta-gonale con un ritra
tto di Dirk  Struan e, sullo sfo
ndo, un clip-per. A nord e ad es
t vi sono dueporte di ferro.": R
ETURN
6512 LET l$="Sei in un corridoio
 dipinto di   azzurro. Il muro a
```

Seguito listato Castello.

```
sud e'in al- cuni punti nero. Se
mbrano trac- ce di fuliggine. Il
 soffitto    ha una vistosa crep
a da cui fil-trano dei raggi di
sole. A ovestc'e'una porta di fe
rro . A est  c'e'una porta rossa
.": RETURN
6513 LET l$="Sei in una stanza s
carsamente    illuminata con dell
e porte rossea nord e ad ovest."
: RETURN
6514 LET l$="Sei in un corridoio
. Da una fe- ritoia ad est posso
 vedere il    bosco. C'e'un passa
ggio a nord  ed una porta rossa
a sud.": RETURN
6515 LET l$="Sei in una stanza d
i forma qua- drata. In un angolo
 c'e'un gros-so tavolo di noce.
Si puo'andarea nord e c'e'una po
rta rossa a  est.": RETURN
6516 LET l$="Sei sulle scale. Il
 soffitto e' una volta a botte.
i muri della parte bassa sono pi
u'umidi che  in cima alla rampa.
": RETURN
6517 LET l$="Sei in una stanza a
mpia. In un  angolo c'e'un gross
o tavolo tut-to tarlato e con gl
i angoli s-  mussati. C'e'una po
rta nera a   ovest e dei passagg
i a nord e   sud.": RETURN
6518 LET l$="Sei nella libreria
del castello.C'e'una porta nera
ad est,una   rossa a sud ed un'a
ltra rossa a ovest.": RETURN
6519 LET l$="Sei nella libreria
del castello.C'e'una porta rossa
 ad ovest,una rossa a sud ed un'
altra rossa a nord.": RETURN
6520 LET l$="Sei nella libreria
del castello.C'e'una porta rossa
 ad ovest ed un'altra rossa a no
rd.": RETURN
6521 LET l$="Sei nella libreria
del castello.C'e'una porta rossa
 ad est ed unaltra rossa a nord.
": RETURN
6522 LET l$="Sei nella libreria
del castello.C'e'una porta rossa
 ad est,una  rossa a sud ed un'a
ltra rossa a nord.": RETURN
6523 LET l$="Sei nella libreria
del castello.C'e'una porta rossa
 ad est ed unaltra rossa a sud."
: RETURN
6524 LET l$="Sei sulle scale. Il
 soffitto e' una volta a botte.
I muri sono  tutti umidi.": RETU
RN
6525 LET l$="Sei in una stanza s
cavata nella roccia.Il soffitto
e'piu'basso  che nei locali supe
riori. C'e'  una porta rossa a o
vest, un pas-saggio a est ed uno
 a sud": RETURN
6526 LET l$="Sei nella cantina d
el castello. Su un blocco di pie
tra a forma  di cubo c'e'un bott
one rosso.": RETURN
6527 LET l$="Sei in un corridoio
 scavato nel-la roccia. Il muro
a nord e'    molto umido. Ci son
o due passag-gi: uno a est e l'a
ltro ad ovest": RETURN
6528 LET l$="Sei in quella che d
oveva essere la stanza delle tor
ture. C'e'un passaggio a sud e u
no ad ovest. "+("C'e'un gobbo se
duto su un tavolo" AND gf): RETURN
6529 LET l$="Sei in un corridoio
 con un pas- saggio a nord ed un
a porta rossaa sud.": RETURN
6530 LET l$="Sei in una stanza i
n cui l'unicauscita e'una porta
rossa a nord.Il pavimento trema
ad ogni passoe sul soffitto c'e'
una botola dacui si infiltra del
l'acqua che  cola sul muro. La b
otola sembra molto precaria.": R
ETURN
6600 REM ■accetta mossa■
6605 LET v$="": LET b$="": LET c
$=""
6610 INPUT "COSA FACCIO ?"; LINE
 i$
6620 IF i$="" THEN GO TO 6610
6625 IF i$(1)=" " THEN GO TO 6610
6630 LET pun=1: LET v$=""
6640 LET v$=v$+i$(pun)
6645 LET pun=pun+1
6650 IF pun>LEN i$ THEN GO TO 6660
6655 IF i$(pun)<>" " THEN GO TO
6640
6660 IF LEN i$-pun<3 THEN LET pa
r=1: RETURN
6670 LET pun=pun+1: LET fla=0
6680 LET b$=b$+i$(pun)
6685 LET pun=pun+1
6690 IF LEN i$<pun THEN LET par=
2: RETURN
6700 IF fla AND i$(pun)=" " THEN
 GO TO 6720
6705 IF i$(pun)=" " THEN LET fla
=1
6710 GO TO 6680
6720 IF LEN i$-pun<1 THEN LET pa
r=2: RETURN
6730 LET c$=i$(pun+1 TO )
6740 LET par=3: RETURN
6800 REM ■ presentazione■
6810 PRINT '"Ti trovi in un cast
ello abbando-nato. Devi riuscire
 ad uscirne  sano e salvo affron
tando i vari pericoli che si ann
idano tra le mura."
6820 PRINT '"Ti consiglio di far
e molta at-  tenzione. HA! HA! H
A!..."
6830 INPUT "Premi ENTER per comi
nciare";i$
6899 RETURN
6900 REM ■uscito■
6910 PRINT "Ben fatto sei riusci
to a comple-tare l'avventura.Non
 ti resta    che attraversare il
sentiero delparco ed andartene a
lla prima    fermata dell'autobus
 senza fartiinvestire."
6999 RETURN
7000 REM ■ oggetti ■
7010 LET uno=0
7020 FOR g=1 TO 13
7030 IF l(g)<>luogo THEN GO TO 7
050
7040 IF NOT uno THEN PRINT '"Ved
o di interessante :": LET uno=1
7045 PRINT o$(g)
7050 NEXT g
7060 RETURN
9990 REM ■ FINE ■
9991 INPUT " ancora o fine (a/f)
?"; LINE i$
9992 IF i$="a" THEN RUN
9993 IF i$<>"f" THEN GO TO 9991
9994 STOP
```

SERVIZIO SOFTWARE

PERSONAL SOFTWARE



P.S. propone ai propri lettori i dischi o le cassette dei programmi pubblicati. I programmi, provati e garantiti, sono di immediato utilizzo.

P.S. n°	Programma	Sistema	Prezzo	Codice	Supporto
3	La carta del cielo Collisione	Apple II	30.000	1	Disco
4	Interi in precisione multipla Grafica 3D	Apple II	40.000	4	Disco
5	Pretty printer Shape table	Apple II	30.000	6	Disco
7	Data base modulare	Apple II	25.000	7	Disco
12-13	Wei-ch'i	CBM 3032	20.000	8	Cassetta
14	Tool-Kit	C 64	35.000	9	Cassetta
19	Type Writer	VIC 20	30.000	10	Disco
20	Scopa	C 64 - 3032	25.000	11	Cassetta

Per richiedere i programmi in contrassegno, pagando direttamente al postino la cifra indicata, inviare il seguente tagliando

Spedire in busta chiusa a Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Inviatemi i seguenti nastri e/o dischi con i programmi pubblicati su P.S.

GRUPPO EDITORIALE JACKSON

Cod. ________ **a L.** ________

Cod. ________ **a L.** ________

Cod. ________ **a L.** ________

Cod. ________ **a L.** ________

Cod. ________ **a L.** ________

+ SPESE POSTALI (contributo fisso) L. 3.000 **TOTALE L.** ________

che pagherò al postino alla consegna del pacco.

Cognome ....

Nome ....

Indirizzo ....

CAP ....

Città ....

Firma ....

PERSONAL SOFTWARE

Roulette

Un classico gioco per il vostro Texas TI99

di *Mauro Cristuib Grizzi*

Questo programma è stato scritto per il TI99/4A fornito di modulo SSS Extended BASIC, e permette di giocare alla roulette europea (quella diffusa nei nostri Casinò e senza il doppio zero).
Possono esserci fino a sei giocatori contemporaneamente, mentre le giocate possibili sono le seguenti:

- Numeri Pieni: è la classica giocata su un numero che, se estratto, rende 35 volte la posta;
- Cavalli: gruppi di due numeri, ad esempio 1 e 2 oppure 1 e 4; paga 17 volte la posta;
- Terzine: gruppi di tre numeri consecutivi sulla stessa riga, come 1,2,3 oppure 13,14,15; paga 11 volte la posta;
- Carré: gruppi di quattro numeri in quadrato, come 1, 2, 3, 5 oppure 14, 15, 17, 18; paga 8 volte la posta;
- Sestine: gruppi di sei numeri consecutivi su due righe, come sestina 1-6, oppure sestina 13-18; paga 5 volte la posta;
- Dozzine: le tre dozzine giocabili sono la 1-12, la 13-24 e la 25-36; pagano 2 volte la posta;
- Colonne: le tre colonne A, B, e C pagano 2 volte la posta;
- Pari/Dispari e Rosso/Nero pagano 1 volta la posta.

Il programma, dopo essersi informato sul numero dei giocatori e sui rispettivi capitali disponibili, passa al primo giocatore e gli chiede quali giocate intenda fare tramite un menu di gioco. Scegliendo la (o le) opzioni desiderate, si visualizzerà il tavolo verde ed il programma chiederà numeri e puntata. Per puntate su

Listato 1. *Il listato del programma Roulette, è stato ottenuto usando una speciale codifica per i caratteri di controllo: quando trovate in una stringa un carattere sottolineato, ciò significa che dovrete premere quel tasto insieme al tasto Control. Ad esempio, A significa Control + A.*

```
100 GOTO 130 :: OPTION BASE
1 :: DIM M(34,6),MA(34,6),M2
(6,6),PU(7),FP(6),NOME$(6),C
L$(6),COL$(3,6):: CALL CHARS
ET :: CALL GCHAR
110 CALL CLEAR :: CALL SCREE
N :: CALL COLOR :: CALL SOUN
D :: CALL KEY :: CALL CHAR :
: CALL CHARPAT :: CALL VCHAR
 :: CALL HCHAR :: RANDOMIZE

120 PLAYER,PL,GI,I,W,N,A,X,K
EY,STATUS,A$,B$,C$ :: !@P-
130 GOSUB 1390 :: CALL CLEAR
 :: CALL CHARSET
140 FOR I=49 TO 56 :: CALL C
HARPAT(I,A$):: CALL CHAR(I+6
3,A$,I+71,A$):: NEXT I :: CA
LL CHARPAT(48,A$,57,B$):: CA
LL CHAR(102,A$,103,B$,110,A$
,111,B$)
150 CALL COLOR(0,16,1,1,8,1,
7,7,15,9,7,15,10,2,15,11,7,1
5,12,2,15,14,8,15)
160 CALL CHAR(130,"000000000
0000000808080808080800000
0FF"):: CALL CHAR(136,"00000
0FF")
170 RANDOMIZE :: FOR I=5 TO
8 :: CALL COLOR(I,8,1):: NEX
T I :: FOR I=2 TO 4 :: CALL
COLOR(I,16,1):: NEXT I
180 PRINT "HHHHHHHHHHHHHHHHH
HHHHHHHHHHH": :" INSERIRE LE
 GIOCATE,TRANNE  LA PUGLIA,I
N UNITA'DI 1000": :"HHHHHHHH
HHHHHHHHHHHHHHHHHHHH": : : :
: : : : : : : : : : : :
: : : :
190 INPUT "QUANTI GIOCATORI?
(MAX.6)":PLAYER :: FOR PL=1
TO PLAYER :: INPUT "NOME,PRE
GO? ":NOME$(PL):: INPUT "PUG
LIA DI PARTENZA?":PU(PL):: P
RINT :: NEXT PL :: G
I=1
200 FOR PL=1 TO PLAYER :: IF
 PL=FP(PL)THEN 500
210 PU(7)=0 :: CL$(PL)="" ::
 FOR I=1 TO 3 :: COL$(I,PL)=
"" :: NEXT I :: FOR I=1 TO 3
4
220 MA(I,PL),M(I,PL)=0 :: NE
XT I :: FOR I=1 TO 6 :: M2(I
,PL)=0 :: NEXT I :: GOSUB 13
30
230 CALL KEY(0,KEY,STATUS)::
 IF STATUS=0 OR(KEY<48 OR KE
Y>57)THEN 230
240 KEY=KEY-48 :: IF KEY=0 T
HEN 470
250 ON KEY GOSUB 270,290,310
,330,350,380,410,430,450
260 GOSUB 1330 :: GOTO 230
270 GOSUB 1110 :: INPUT "QUA
NTI PIENI?(MAX.6) ":N :: FOR
 I=1 TO N :: INPUT "NUMERO?P
UNTATA?":M(I,PL),A :: MA(I,P
L)=A*1000
280 PU(7)=PU(7)+MA(I,PL):: N
EXT I :: RETURN
290 GOSUB 1110 :: INPUT "QUA
NTI CAVALLI?(MAX.6) ":N :: F
OR I=1 TO N :: INPUT "NUMERI
?PUNTATA?":M(I+6,PL),M2(I,PL
),A
300 MA(I+6,PL)=A*1000 :: PU(
7)=PU(7)+MA(I+6,PL):: NEXT I
 :: RETURN
310 GOSUB 1110 :: INPUT "QUA
NTI CARRE'?(MAX.6) ":N :: FO
R I=1 TO N :: INPUT "NUM.PIU
'BASSO?PUNTATA?":M(I+12,PL),
A
320 MA(I+12,PL)=A*1000 :: PU
(7)=PU(7)+MA(I+12,PL):: NEXT
 I :: RETURN
330 GOSUB 1110 :: INPUT "QUA
NTE TERZINE?(MAX.6) ":N :: F
OR I=1 TO N :: INPUT "NUM.PI
U'BASSO?PUNTATA?":M(I+18,PL)
,A :: MA(I+18,PL)=A*1000
340 PU(7)=PU(7)+MA(I+18,PL):
: NEXT I :: RETURN
```

Terzine, Carré, Sestine e Dozzine sarà sufficiente battere il numero più basso compreso nel gruppo, ed il computer provvederà a calcolare anche gli altri automaticamente. Dopo ogni puntata viene visualizzato il capitale ancora disponibile e quello giocato; finito di puntare, occorre premere il tasto di fine giocata ed il calcolatore, dopo avervi informato sul totale giocato, passerà a raccogliere le puntate del giocatore seguente.

Quando tutti avranno puntato, verrà estratto il fatidico numero e visualizzati gli esiti, giocatore per giocatore e giocata per giocata. Si passerà quindi a raccogliere le puntate successive.

Il programma è ricco di controlli e quindi, se provate a barare, vi prenderà sempre in "castagna". Se poi malauguratamente doveste finire i soldi, il computer ve ne presterà quanti ne vorrete, e senza pretenderne la restituzione!

Alcuni commenti al listato

100-120 - Elenco dei sottoprogrammi e delle variabili usate, per ridurre il tempo di attesa dopo il Run.

130-160 - Presentazione e definizione caratteri speciali.

170-190 - Raccolta dei dati su numero di giocatori, rispettivi nomi e capitali disponibili.

200-260 - Visualizzazione del menu di gioco ed accettazione della opzione selezionata.

270-280 - Subroutine accettazione Numeri Pieni.

290-300 - Subroutine accettazione Cavalli.

310-320 - Subroutine accettazione Carré.

330-340 - Subroutine accettazione Terzine.

```
350 GOSUB 1110 :: INPUT "QUA
NTE SESTINE?(MAX.6)":N :: FO
R I=1 TO N :: INPUT "NUM.PIU
'BASSO?PUNTATA?":M(I+24,PL),
A :: FOR W=1 TO 34 STEP 3 ::
 IF M(I+24,PL)=W THE
N 370 ELSE 360
360 NEXT W :: PRINT "NUMERO
NON CONGRUENTE" :: FOR W=1 T
O 150 :: NEXT W :: GOTO 350

370 MA(I+24,PL)=A*1000 :: PU
(7)=PU(7)+MA(I+24,PL):: NEXT
 I :: RETURN
380 GOSUB 1110 :: INPUT "QUA
NTE DOZZINE?":N :: FOR I=1 T
O N :: INPUT "NUM.PIU'BASSO?
PUNTATA?":M(I+28,PL),A :: FO
R W=1 TO 25 STEP 12 :: IF M(
I+28,PL)=W THEN 400
ELSE 390
390 NEXT W :: PRINT "NUMERO
NON CONGRUENTE" :: FOR W=1 T
O 150 :: NEXT W :: GOTO 380

400 MA(I+28,PL)=A*1000 :: PU
(7)=PU(7)+MA(I+28,PL):: NEXT
 I :: RETURN
410 CALL CLEAR :: INPUT "QUA
NTE COLONNE?":N :: PRINT ::
FOR I=1 TO N :: INPUT "COLON
NA?PUNTATA?":COL$(I,PL),A ::
 MA(I+30,PL)=A*1000
420 PU(7)=PU(7)+MA(I+30,PL):
: NEXT I :: RETURN
430 CALL CLEAR :: INPUT "PAR
I-DISPARI?(P/D)PUNTATA?  ":A
$,A :: IF A$="P".THEN COL$(3
,PL)="PARI" ELSE COL$(3,PL)=
"DISPARI"
440 MA(33,PL)=A*1000 :: PU(7
)=PU(7)+MA(33,PL):: RETURN
450 CALL CLEAR :: INPUT "COL
ORE (R/N)?PUNTATA?":A$,A ::
IF A$="R" THEN CL$(PL)="ROSS
O" ELSE CL$(PL)="NERO"
460 MA(34,PL)=A*1000 :: PU(7
)=PU(7)+MA(34,PL):: RETURN
470 DISPLAY AT(24,1):USING "
TOT GIOCATO        #######":PU(
7)
480 FOR W=1 TO 500 :: NEXT W
 :: IF PU(7)<=PU(PL)THEN 500

490 CALL SOUND(100,500,0)::
PRINT "GIOCATA SUPERIORE A Q
UANTO  POSSEDUTO----NON ACCE
TTATA" :: FOR I=1 TO 500 ::
NEXT I :: GOTO 210
500 NEXT PL :: PRINT : : ::
X=INT(37*RND):: CALL CLEAR :
: FOR I=1 TO 10 :: DISPLAY A
T(16,5)BEEP:"ATTENDERE PREGO
..."
510 FOR W=1 TO 50 :: NEXT W
:: CALL CLEAR :: FOR W=1 TO
50 :: NEXT W :: NEXT I
520 FOR I=1 TO 35 STEP 2 ::
IF X=I THEN B$="NERO" :: GOT
O 540
530 NEXT I :: IF X=0 THEN B$
="---" ELSE B$="ROSSO"
540 FOR I=2 TO 36 STEP 2 ::
IF X=I THEN C$="PARI" :: GOT
O 560
550 NEXT I :: IF X=0 THEN C$
="---" ELSE C$="DISP."
560 FOR PL=1 TO PLAYER :: PU
(7)=0 :: IF PL=FP(PL)THEN 10
80
570 PRINT "USCITO IL";X;"-"&
C$&"-"&B$&"-": : :: CALL SOU
ND(200,1500,0):: PRINT "RISU
LTATI DI ";NOME$(PL): : :: F
OR I=1 TO 6 :: IF MA(I,PL)=0
 THEN 610
580 IF X<>M(I,PL)THEN 600
590 PRINT "PIENO   ";M(I,PL)
;TAB(23);"*VINCE" :: PU(7)=P
U(7)+35*MA(I,PL):: GOTO 610

600 PRINT "PIENO   ";M(I,PL)
;TAB(23);"PERDE" :: PU(7)=PU
(7)-MA(I,PL)
610 NEXT I :: FOR I=7 TO 12
:: IF MA(I,PL)=0 THEN 650
620 IF X=M(I,PL)OR X=M2(I-6,
PL)THEN 640

630 PRINT "CAVALLO ";M(I,PL)
;M2(I-6,PL);TAB(23);"PERDE"
:: PU(7)=PU(7)-MA(I,PL):: GO
TO 650
640 PRINT "CAVALLO ";M(I,PL)
;M2(I-6,PL);TAB(23);"*VINCE"
 :: PU(7)=PU(7)+17*MA(I,PL)

650 NEXT I :: FOR I=13 TO 18
 :: IF MA(I,PL)=0 THEN 690
660 A=M(I,PL):: IF X=A OR X=
A+1 OR X=A+3 OR X=A+4 THEN 6
80
```

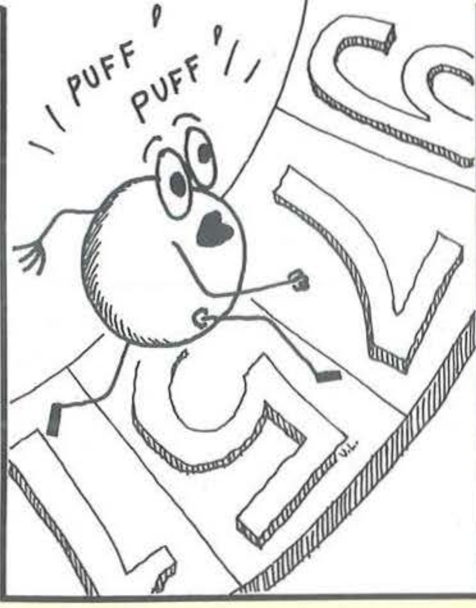
Seguito listato Roulette.

```
670 PRINT "CARRE' ";A;A+1;A+
3;A+4;TAB(23);"PERDE" :: PU(
7)=PU(7)-MA(I,PL):: GOTO 690

680 PRINT "CARRE' ";A;A+1;A+
3;A+4;TAB(23);"*VINCE" :: PU
(7)=PU(7)+8*MA(I,PL)
690 NEXT I :: FOR I=31 TO 32
 :: IF MA(I,PL)=0 THEN 760
700 FOR W=1 TO 34 STEP 3 ::
IF W=X THEN A$="A"
710 IF X=W+1 THEN A$="B"
720 IF X=W+2 THEN A$="C"
730 NEXT W :: IF A$=COL$(I-3
0,PL)THEN 750
740 PRINT "COLONNA       ";CO
L$(I-30,PL);TAB(23);"PERDE"
:: PU(7)=PU(7)-MA(I,PL):: GO
TO 760
750 PRINT "COLONNA       ";CO
L$(I-30,PL);TAB(23);"*VINCE"
 :: PU(7)=PU(7)+2*MA(I,PL)
760 NEXT I :: FOR I=19 TO 24
 :: IF MA(I,PL)=0 THEN 800
770 A=M(I,PL):: IF X=A OR X=
A+1 OR X=A+2 THEN 790
780 PRINT "TERZINA ";A;A+1;A
+2;TAB(23);"PERDE" :: PU(7)=
PU(7)-MA(I,PL):: GOTO 800
790 PRINT "TERZINA ";A;A+1;A
+2;TAB(23);"*VINCE" :: PU(7)
=PU(7)+11*MA(I,PL)
800 NEXT I :: FOR I=25 TO 28
 :: IF MA(I,PL)=0 THEN 840
810 FOR W=M(I,PL)TO M(I,PL)+
5 :: IF X<>W THEN 830
820 PRINT "SESTINA ";W;"-";W
+5;TAB(23);"*VINCE" :: PU(7)
=PU(7)+5*MA(I,PL):: GOTO 840

830 NEXT W :: PRINT "SESTINA
 ";M(I,PL);"-";M(I,PL)+5;TAB
(23);"PERDE" :: PU(7)=PU(7)-
MA(I,PL)
840 NEXT I :: FOR I=29 TO 30
 :: IF MA(I,PL)=0 THEN 880
850 IF X<M(I,PL)OR X>M(I,PL)
+11 THEN 870
860 PRINT "DOZZINA ";M(I,PL)
;"-";M(I,PL)+11;TAB(23);"*VI
NCE" :: PU(7)=PU(7)+2*MA(I,P
L):: GOTO 880
870 PRINT "DOZZINA ";M(I,PL)
;"-";M(I,PL)+11;TAB(23);"PER
DE" :: PU(7)=PU(7)-MA(I,PL)

880 NEXT I :: IF MA(33,PL)=0
 THEN 970
890 IF X=0 THEN 960
900 IF COL$(3,PL)="PARI" THE
N A=1 ELSE A=2
```




```
910 FOR I=2 TO 36 STEP 2 ::
IF X=I THEN 920 ELSE 930
920 ON A GOTO 950,960
930 NEXT I
940 ON A GOTO 960,950
950 PRINT COL$(3,PL);TAB(23)
;"*VINCE" :: PU(7)=PU(7)+MA(
33,PL):: GOTO 970
960 PRINT COL$(3,PL);TAB(23)
;"PERDE" :: PU(7)=PU(7)-MA(3
3,PL)
970 IF MA(34,PL)=0 THEN 1010

980 FOR I=1 TO 35 STEP 2 ::
IF X=I AND CL$(PL)="NERO" TH
EN PRINT CL$(PL);TAB(23);"*V
INCE" :: PU(7)=PU(7)+MA(34,P
L):: GOTO 1010
990 NEXT I :: FOR I=2 TO 36
STEP 2 :: IF X=I AND CL$(PL)
="ROSSO" THEN PRINT CL$(PL);
TAB(23);"*VINCE" :: PU(7)=PU
(7)+MA(34,PL):: GOTO 1010
1000 NEXT I :: PRINT CL$(PL)
;TAB(23);"PERDE" :: PU(7)=PU
(7)-MA(34,PL)
1010 PU(PL)=PU(PL)+PU(7)
1020 PRINT USING "ESITO GIOC
ATA +##########":PU(7):: PRI
NT USING "NUOVA PUGLIA  ####
#######":PU(PL):: IF PU(PL)>
0 THEN 1060
1030 CALL SOUND(1000,140,0):
: PRINT : :"HAI FINITO I SOL
DI!": : :: INPUT "VUOI UNA N
UOVA PUGLIA? ":A$ :: IF A$<>
"SI" THEN 1050
1040 INPUT "QUANTO? ":PU(PL)
:: GOTO 1060
1050 FP(PL)=PL
1060 PRINT : :"****** PREMI
UN TASTO ******": : :
1070 CALL KEY(0,KEY,STATUS):
: IF STATUS=0 THEN 1070 ELSE
 1080
1080 NEXT PL
1090 GI=GI+1 :: GOTO 200
1100 END
1110 CALL CLEAR :: CALL COLO
R(13,2,15)
1120 FOR I=5 TO 14 :: CALL V
CHAR(8,I,130,16):: NEXT I ::
 FOR I=19 TO 28 :: CALL VCHA
R(10,I,130,14):: NEXT I
1130 FOR I=8 TO 11 STEP 3 ::
 CALL VCHAR(9,I,131,15):: NE
XT I
1140 FOR I=22 TO 25 STEP 3 :
: CALL VCHAR(10,I,131,14)::
NEXT I
1150 FOR I=10 TO 22 STEP 2 :
: CALL HCHAR(I,6,132,8):: CA
LL HCHAR(I,20,132,8):: NEXT
I :: CALL HCHAR(18,6,136,8):
: CALL HCHAR(14,20,136,8)
1160 DISPLAY AT(9,4)SIZE(-8)
:"BBBBnBBB" :: DISPLAY AT(11
,4)SIZE(-2):"Bx"
1170 DISPLAY AT(11,7)SIZE(-2
):"Bq" :: DISPLAY AT(11,10)S
IZE(-2):"Bz" :: DISPLAY AT(1
3,4)SIZE(-2):"Bs"
1180 DISPLAY AT(13,7)SIZE(-2
):"B|" :: DISPLAY AT(13,10)S
IZE(-2):"Bu" :: DISPLAY AT(1
5,4)SIZE(-2):"B~"
1190 DISPLAY AT(15,7)SIZE(-2
):"Bw" :: DISPLAY AT(15,10)S
IZE(-2):"Bo" :: DISPLAY AT(1
7,4)SIZE(-2):"pf"
1200 DISPLAY AT(17,7)SIZE(-2
):"xx" :: DISPLAY AT(17,10)S
IZE(-2):"pq"
1210 DISPLAY AT(19,4)SIZE(-2
):"xz" :: DISPLAY AT(19,7)SI
ZE(-2):"ps" :: DISPLAY AT(19
,10)SIZE(-2):"x|"
1220 DISPLAY AT(21,4)SIZE(-2
):"pu" :: DISPLAY AT(21,7)SI
ZE(-2):"x~" :: DISPLAY AT(21
,10)SIZE(-2):"pw"
1230 DISPLAY AT(23,4)SIZE(-2
):"BA" :: DISPLAY AT(23,7)SI
ZE(-2):"BB" :: DISPLAY AT(23
,10)SIZE(-2):"BC"
1240 DISPLAY AT(11,18)SIZE(-
2):"xo" :: DISPLAY AT(11,21)
SIZE(-2):"qf"
1250 DISPLAY AT(11,24)SIZE(-
2):"yx" :: DISPLAY AT(13,18)
SIZE(-2):"qq" :: DISPLAY AT(
```

```
13,21)SIZE(-2):"yz"
1260 DISPLAY AT(13,24)SIZE(-
2):"qs" :: DISPLAY AT(15,18)
SIZE(-2):"y|"
1270 DISPLAY AT(15,21)SIZE(-
2):"qu" :: DISPLAY AT(15,24)
SIZE(-2):"y~" :: DISPLAY AT(
17,18)SIZE(-2):"qw"
1280 DISPLAY AT(17,21)SIZE(-
2):"yo" :: DISPLAY AT(17,24)
SIZE(-2):"rf"
1290 DISPLAY AT(19,18)SIZE(-
2):"zx" :: DISPLAY AT(19,21)
SIZE(-2):"rq" :: DISPLAY AT(
19,24)SIZE(-2):"zz"
1300 DISPLAY AT(21,18)SIZE(-
2):"rs" :: DISPLAY AT(21,21)
SIZE(-2):"z|" :: DISPLAY AT(
21,24)SIZE(-2):"ru"
1310 DISPLAY AT(23,18)SIZE(-
2):"BA" :: DISPLAY AT(23,21)
SIZE(-2):"BB" :: DISPLAY AT(
23,24)SIZE(-2):"BC"
1320 PRINT :: RETURN
1330 CALL CLEAR :: CALL COLO
R(13,9,2):: DISPLAY AT(1,1):
"GIOCATA N.";GI;"DI";" "&NOM
E$(PL):: DISPLAY AT(2,1):"DD
DDDDDDDDDDDDDDDDDDDDDDDDDD"

1340 DISPLAY AT(3,1):"PRESS
1 PER PIENI" :: DISPLAY AT(5
,1):"PRESS 2 PER CAVALLI"
1350 DISPLAY AT(7,1):"PRESS
3 PER CARRE'" :: DISPLAY AT(
9,1):"PRESS 4 PER TERZINE" :
: DISPLAY AT(11,1):"PRESS 5
PER SESTINE" :: DISPLAY AT(1
3,1):"PRESS 6 PER DO
ZZINE"
1360 DISPLAY AT(15,1):"PRESS
 7 PER COLONNE" :: DISPLAY A
T(17,1):"PRESS 8 PER PARI/DI
SP." :: DISPLAY AT(19,1):"PR
ESS 9 PER ROSSO/NERO"
1370 DISPLAY AT(21,1):"PRESS
 0 PER FINE" :: DISPLAY AT(2
3,1):"POSSEDUTE" :: DISPLAY
AT(23,17):USING "#######":PU
(PL)
1380 DISPLAY AT(24,1):"DISP.
DA GIOCARE" :: DISPLAY AT(24
,17):USING "#######":PU(PL)-
PU(7):: RETURN
1390 FOR I=65 TO 88 :: CALL
CHARPAT(I,A$):: CALL CHAR(I+
31,A$):: NEXT I :: FOR I=9 T
O 11 :: CALL COLOR(I,2,11)::
 NEXT I
1400 CALL CLEAR :: CALL SCRE
EN(2):: CALL CHAR(136,""):: 
CALL COLOR(14,9,9):: FOR I=2
 TO 32 :: CALL HCHAR(1,I,136
):: NEXT I :: FOR I=1 TO 24
:: CALL VCHAR(I,32,1
36)
1410 NEXT I :: FOR I=32 TO 2
 STEP -1 :: CALL HCHAR(24,I,
136):: NEXT I :: FOR I=24 TO
 1 STEP -1 :: CALL VCHAR(I,2
,136):: NEXT I :: CALL CHAR(
129,""):: CALL COLOR
(13,5,5):: W=3 :: PL=13 :: X
=5 :: N=29
1420 FOR I=X TO N :: CALL HC
HAR(W,I,129):: NEXT I :: FOR
 A=W TO PL :: CALL VCHAR(A,N
,129):: NEXT A :: FOR I=N TO
 X STEP -1 :: CALL HCHAR(PL,
I,129)
1430 NEXT I :: FOR A=PL TO W
 STEP -1 :: CALL VCHAR(A,X,1
29):: NEXT A :: X=X+1 :: N=N
-1 :: W=W+1 :: PL=PL-1 :: IF
 PL-W<0 THEN 1440 ELSE 1420

1440 FOR W=3 TO 8 :: CALL CO
LOR(W,16,5):: NEXT W :: A$="
0613ROULETTE" :: GOSUB 1540

1450 A$="0810BYAMACRISTUIB"
:: GOSUB 1540 :: CALL CHAR(1
20,""):: CALL COLOR(12,11,11
):: CALL CHAR(58,"FFFFFFFFFF
FFFFFF")
1460 FOR I=17 TO 19 STEP 2 :
: FOR W=8 TO 26 :: CALL HCHA
R(I,W,120):: NEXT W :: FOR W
=26 TO 8 STEP -1 :: CALL HCH
AR(I+1,W,120):: NEXT W :: NE
XT I
1470 A$="1809sdw`rxhmrsqtldm
sr" :: GOSUB 1540 :: A$="191
3bnlotsdq" :: GOSUB 1540 ::
FOR A=24 TO 1 STEP -2 :: FOR
 X=2 TO 32 :: CALL GCHAR(A,X
,W):: IF W=32 THEN C
ALL HCHAR(A,X,58)
1480 IF W=129 THEN X=29
1490 IF W=120 THEN X=26
1500 NEXT X :: FOR X=32 TO 2
 STEP -1 :: CALL GCHAR(A-1,X
,W):: IF W=32 THEN CALL HCHA
R(A-1,X,58)
1510 IF W=129 THEN X=5
1520 IF W=120 THEN X=8
1530 NEXT X :: NEXT A :: RET
URN
1540 A=VAL(SEG$(A$,1,2)):: X
=VAL(SEG$(A$,3,2)):: FOR I=5
 TO LEN(A$):: W=ASC(SEG$(A$,
I,1)):: CALL HCHAR(A,X,W)::
CALL SOUND(5,800,0):: X=X+1
:: NEXT I :: RETURN
```

350-370 - Subroutine accettazione Sestine e controllo correttezza dei numeri scelti.
380-400 - Subroutine accettazione Dozzine e controllo correttezza dei numeri scelti.
410-420 - Subroutine accettazione Colonne.
430-440 - Subroutine accettazione Pari/Dispari.
450-460 - Subroutine accettazione Rosso/Nero.
470-490 - Visualizzazione del totale giocato e controllo che questo non superi il capitale disponibile.
500-510 - Ciclo di attesa prima dell'estrazione del numero vincente.
520-550 - Calcolo del colore del numero estratto e verifica della sua divisibilità per due (Pari o Dispari).
560 - Ciclo For/Next per il numero totale di giocatori, con controllo sui giocatori che hanno esaurito il proprio capitale e vanno quindi "saltati" dal ciclo stesso.
570 - Presentazione del numero uscito ed intestazione dei risultati della giocata del giocatore PL.
580-600 - Analisi della giocata sui Pieni.
610-650 - Analisi della giocata sui Cavalli.
660-690 - Analisi della giocata sui Carré.
700-760 - Analisi della giocata sulle Colonne.
770-800 - Analisi della giocata sulle Terzine.
810-840 - Analisi della giocata sulle Sestine.
850-880 - Analisi della giocata sulle Dozzine.
890-1010 - Analisi della giocata su Pari/Dispari e Rosso/Nero.
1020-1080 - Calcolo e visualizzazione di esito giocata, nuovo capitale e, nel caso questo sia nullo, offerta di una nuova puglia.
1110-1320 - Subroutine visualizzazione del tavolo di gioco.
1330-1380 - Subroutine visualizzazione menu di gioco.
1390-1540 - Subroutine presentazione del gioco. ■

One Touch per VIC 20

Un modo per velocizzare l'input dei programmi

di *Giorgio Bellegatti*

Come è noto, il VIC 20 e i sistemi Commodore in genere offrono il vantaggio di poter abbreviare diverse istruzioni BASIC.
Nonostante questo, però, ho pensato di rendere ancor meno pesante la battitura dei programmi, riducendo alla pressione di un solo tasto (+ Ctrl) la scrittura di interi comandi.
Inoltre, a differenza delle abbreviazioni standard, ho attribuito alle varie istruzioni il massimo dei caratteri compatibili, per esempio l'apertura delle parentesi dopo tutte le funzioni il Return dopo Run, List, Cont.
Nell'assegnazione dei tasti, come illustrato nella tabella 1, ho cercato di raggruppare le istruzioni in modo omogeneo sulla tastiera e precisamente, nella parte alta (prime 2 file di tasti) ho assegnato i comandi ed, in particolare, addensati alla sinistra quelli di uso più frequente (ciò diventa comodo tenendo conto che per attivare un'istruzione si deve premere contemporaneamente anche il tasto Ctrl), mentre, nella parte inferiore, la penultima riga contiene le funzioni matematiche e l'ultima le funzioni stringa.
Inoltre sono stati attivati anche i tasti funzione (F1, F3, F5, F7).
Nella scelta delle istruzioni da abilitare ho tralasciato quelle di due lettere in quanto mi sembrava che il vantaggio fosse irrilevante.
Esaminando la tabella noterete subito la mancanza di Print, ma per quest'ultima penso che sia già molto comoda l'abbreviazione usuale (infatti, non è persa la possibilità di servirsi anche delle solite abbreviazioni).
Tutte le altre istruzioni sono state abilitate, tranne qualcuna di uso assai raro come Let, Stop, Cmd.
Infine, ho volutamente tralasciato il New al fine di evitare spiacevoli conseguenze alla pressione di un solo tasto errato...
Per rendere, comunque, agevole ricordare la corrispondenza tra i vari tasti e le relative istruzioni, almeno per i primi tempi, è possibile applicare adesivi riportanti le istruzioni su ogni tasto.

TABELLA DI CORRISPONDENZA TASTO-COMANDO (parte superiore tastiera)

Tasto	Comando
←	THEN
1	PEEK(
2	POKE
3	WAIT
4	READ
5	DATA
6	RESTORE
7	CLR
8	END
9	OPEN
0	CLOSE
+	LOAD
—	SAVE
£	VERIFY
Q	GOSUB
W	RETURN
E	GOTO
R	SYS
T	FOR
Y	NEXT
U	STEP
I	GET
O	INPUT
P	DEFFN
@	DIM
*	NOT
↑	AND

TABELLA DI CORRISPONDENZA TASTO-COMANDO (parte inferiore tastiera)

Tasto	Comando
A	ABS(
S	EXP(
D	LOG(
F	SGN(
G	SQR(
H	INT(
J	RND(
K	SIN(
L	COS(
:	TAN(
;	ATN(
=	FRE(0)
Z	ASC(
X	CHR$(
C	LEFT$(
V	RIGHT$(
B	MID$(
N	STR$(
M	VAL(
,	LEN(
.	SPC(
/	TAB(
F1	RUN←
F3	LIST←
F5	CONT←
F7	REM

Tabella 1. *Lista dei tasti che consentono di abbreviare i comandi.*

PERSONAL SOFTWARE

One Touch per VIC 20

Listato 1. *Il programma One Touch.*

```
10 REM*****************
15 REM*****************
20 REM***ONE TOUCH****
25 REM***** BY *******
30 REM****GIORGIO*****
35 REM**BELLEGOTTI****
40 REM*****************
45 REM*****************
50 PRINT"[<1CLR>][<1RED>][<2CRSR D>]*****
*ONE  TOUCH******":PRINT"[<2CRSR D>][<1BL
U>]ATTENDI ALCUNI ISTANTI"
55 P=PEEK(56):PP=P*256
60 FORX=PP-564TOPP-9:READA:IFA<0THENA=P+A
+1
65 POKEX,A:NEXT
70 FORX=700TO722:READA:IFA<0THENA=P+A+1
75 POKEX,A:NEXT:SYS700
80 POKE56,P-3:POKE52,P-3:POKE55,200:POKE5
1,200
85 PRINT"[<3CRSR D>] ONE TOUCH ATTIVATO !
":NEW
90 REM**PUNTATORI**
95 DATA53,-2,66,-2,78,-2,93,-2,103,-2,116
,-2,128,-2,89,-3,47,-2,233,-3
100 DATA247,-3,1,-2,13,-2,25,-2,37,-2,89,
-3,89,-3,152,-3,164,-3,176,-3
105 DATA188,-3,200,-3,212,-3,89,-3,89,-3,
89,-3,90,-3,105,-3,121,-3,134,-3
110 DATA146,-3,89,-3,89,-3,84,-3,97,-3,11
4,-3,128,-3,140,-3,89,-3,136,-2
115 DATA89,-3,158,-3,170,-3,182,-3,194,-3
,206,-3,218,-3,142,-2,226,-3,241,-3
120 DATA252,-3,7,-2,18,-2,32,-2,42,-2,149
,-2,60,-2,72,-2,84,-2,98,-2
125 DATA109,-2,122,-2,89,-3,156,-2,0,0,0,
0,0,0,0,0
130 REM**COMANDI**
135 DATA20,65,83,67,40,0,20,67,72,82,36,4
0,0,20,76,69,70,84,36,40,0
140 DATA20,82,73,71,72,84,36,40,0,20,77,7
3,68,36,40,0,20,83,84,82,36,40,0
145 DATA20,86,65,76,40,0,20,76,69,78,40,0
,20,83,80,67,40,0,20,84,65,66,40,0
150 DATA20,65,66,83,40,0,20,69,88,80,40,0
,20,76,79,71,40,0,20,83,71,78,40,0
155 DATA20,83,81,82,40,0,20,73,78,84,40,0
,20,82,78,68,40,0,20,83,73,78,40,0
160 DATA20,67,79,83,40,0,20,84,65,78,40,0
,20,65,84,78,40,0,20,70,82,69,40,48,41,0
165 DATA20,71,79,83,85,66,0,20,82,69,84,8
5,82,78,0,20,71,79,84,79,0
170 DATA20,83,89,83,0,20,70,79,82,0,20,78
,69,88,84,0,20,83,84,69,80,0
175 DATA20,71,69,84,0,20,73,78,80,85,84,0
,20,68,69,70,70,78,0,20,68,73,77,0
180 DATA20,78,79,84,0,20,65,78,68,0,20,84
,72,69,78,0,20,80,69,69,75,40,0
185 DATA20,80,79,75,69,0,20,87,65,73,84,0
,20,82,69,65,68,0,20,68,65,84,65,0
190 DATA20,82,69,83,84,79,82,69,0,20,67,7
6,82,0,20,69,78,68,0,20,79,80,69,78,0
195 DATA20,67,76,79,83,69,0,20,76,79,65,6
8,0,20,83,65,86,69,0,20,86,69,82,73,70,89
,0
200 DATA3,82,85,78,13,0,3,76,73,83,84,13,
0,3,67,79,78,84,13,0,3,82,69,77,0,0
205 REM**ROUTINES**
210 DATA173,141,2,201,4,208,46,205,142,2,
240,41,165,203,201,64,240,35,10,168,185,2
04
215 DATA-4,133,1,185,205,-4,133,2,160,0,1
77,1,240,9,153,119,2,230,198,200,76,194,-
2
220 DATA169,4,141,142,2,76,214,235,76,220
,235,0,0,0,0,72,8,165,214,201,20,48,8,32
225 DATA117,233,198,214,76,224,-2,32,135,
229,40,104,76,124,197,0,0
230 DATA120,169,162,141,143,2,169,-2,141,
144,2,169,222,141,4,3,169,-2,141,5,3,88,9
6
```

Il programma, ovviamente in linguaggio macchina, gira con qualsiasi configurazione di memoria ed occupa solo 568 byte.
Il programma BASIC non fa altro che caricare i dati per il linguaggio macchina nella zona più alta della memoria, oltre naturalmente a proteggere questi stessi dati da sovrapposizioni, abbassando i puntatori di fine memoria (55-56).
Appena avete terminato di scrivere, salvate subito il programma senza dare il Run, in quanto esso, terminata l'esecuzione, si autocancella.
Inoltre, state molto attenti a digitare in modo esatto tutti i dati poiché un solo errore può portare al blocco completo del sistema o, comunque, a risultati disastrosi.

Il programma in linguaggio macchina sfrutta la possibilità, offerta dal VIC 20, di intervenire sui puntatori RAM che indirizzano ad alcune routine del sistema operativo.
In particolare ho modificato i puntatori (655-656) alla routine che controlla la tastiera e che viene eseguita 60 volte al secondo, essendo chiamata dall'Irq.
Ciò, però, non può essere effettuato in BASIC con l'istintivo Poke 655,X:Poke 656,Y, che può anche costringervi a spegnere il computer, ma da una breve routine in linguaggio macchina che, prima di modifi-

care questi puntatori, setti il flag I del microprocessore in modo da interrompere temporaneamente le chiamate dell'Irq.
Nel mio programma ciò viene eseguito mediante la Sys700 della riga 75 e risulta anche molto utile poiché, quando si preme Stop-Restore, One Touch viene disabilitato, ed in questo modo, per rimetterlo in funzione, basta digitare in modo diretto una nuova Sys 700.
Con questo sistema, dunque, prima che vengano svolte le normali operazioni del sistema operativo, sono riuscito a far eseguire la mia nuova routine, la quale, in sostanza, non fa altro che controllare se viene premuto un tasto contemporaneamente a Ctrl e, in caso affermativo, a leggere mediante dei puntatori i codici ASCII dell'istruzione corrispondente e a porli nel buffer di tastiera.
Penserà poi il sistema operativo a togliere questi caratteri dal buffer e



a mostrarveli sul video.
Il primo problema che mi si è presentato era quello di riuscire ad abilitare il maggior numero di tasti senza togliere nello stesso tempo nessuna delle loro particolarità.
Infatti, se avessi scelto di far scrivere le varie istruzioni BASIC in seguito alla semplice pressione di un tasto, o insieme a Shift, Ctrl o Commodore, sarebbero venute meno alcune delle sue varie funzioni originali. Ho pensato allora di abilitare il mio One Touch solo nel caso in cui il tasto-istruzione venga premuto per primo e il tasto Ctrl per secondo: il contrario, in pratica, dell'usuale modo di utilizzo dei tasti con Shift o Ctrl.
Così, se per esempio premete i tasti 1 e Ctrl, vedrete visualizzato sul video Peek, ma se premete Ctrl e 1, allora vedrete il cursore diventare di colore nero.

In questo modo sono riuscito ad abilitare ben 53 tasti dei 64 presenti

Listato 2. *Disassemblato della nuova routine che controlla la tastiera (puntata dalle memorie 655-656) nel caso in cui si lavori con l'espansione di memoria da 8 Kbyte.*

```
16290  LDA  653       ; controlla se è stato
16293  CMP  #4        ; premuto il tasto Ctrl:
16295  BNE  16343     ; in caso negativo salta
16297  CMP  654       ; alla normale routine
16300  BEQ  16343     ; del sistema operativo.
16302  LDA  203       ; Controlla se è stato premuto un tasto:
16304  CMP  #64       ; in caso negativo salta alla
16306  BEQ  16343     ; normale routine del sistema operativo.
16308  ASL            ; Legge i puntatori
16309  TAY            ; relativi al tasto
16310  LDA  15820,Y   ; premuto e li pone nelle
16313  STA  1         ; locazioni di memoria
16315  LDA  15821,Y   ; in pagina zero
16318  STA  2         ; 1 e 2.
16320  LDY  #0        ; Legge i codici ASCII
16322  LDA  (1),Y     ; dell'istruzione fino
16324  BEQ  16335     ; al primo zero
16326  STA  631,Y     ; e li memorizza nel
16329  INC  198       ; buffer di tastiera
16331  INY            ; incrementando il
16332  JMP  16322     ; contatore 198.
16335  LDA  #4        ; Memorizza che è stato
16337  STA  654       ; premuto il tasto Ctrl.
16340  JMP  60374     ; Salta alla fine della routine del S.O.
16343  JMP  60380     ; Salta all'inizio della routine del S.O.
```

Listato 3. *Disassemblato della routine che controlla la posizione del cursore sullo schermo (puntata dalle memorie 772-773) nel caso in cui si lavori con l'espansione da 8 Kbyte.*

```
16350  PHA            ; Salva sullo stack l'accumulatore.
16351  PHP            ; Salva il registro di stato.
16352  LDA  214       ; Controlla se il cursore si trova
16354  CMP  #20       ; sulle ultime tre righe dello schermo:
16356  BMI  16366     ; in caso negativo esce dalla routine.
16358  JSR  59765     ; Effettua lo scroll del video.
16361  DEC  214       ; Decrementa la riga del cursore.
16363  JMP  16352     ; Salta ad un nuovo controllo.
16366  JSR  58759     ; Setta la nuova posizione del cursore.
16369  PLP            ; Riprende il registro di stato.
16370  PLA            ; Riprende l'accumulatore.
16371  JMP  50556     ; Salta alla routine originale.
```

sulla tastiera, senza togliere nessuna delle loro varie ed utilissime funzioni: è così possibile, come detto in precedenza, utilizzare anche, qualora sia necessario (per esempio nelle linee di programma più lunghe di 88 caratteri), le abbreviazioni standard.

Ho dovuto, però, aggiungere nella lista dei codici ASCII, all'inizio di ogni istruzione, il codice 20, corrispondente al comando di tastiera Del, in quanto, dovendo il tasto-istruzione essere premuto prima del tasto Ctrl, questo avrebbe scritto anche il segno grafico corrispondente, alterando così la corretta scrittura di un'istruzione.

Infatti, se per esempio premete il tasto Q, vedrete apparire sullo schermo la lettera corrispondente, ma appena premete insieme anche il tasto Ctrl, la Q scomparirà e verrà visualizzato al suo posto il comando Gosub.

Sempre per questo motivo le istruzioni One Touch funzionano anche in modo virgolette, ma non in modo Inst.

Un ultimo problema era dovuto al fatto che quando il cursore giunge in fondo allo schermo e, quindi, avviene lo scroll del video, i caratteri immagazzinati nel buffer di tastiera vengono persi e di conseguenza, la scrittura delle istruzioni non avveniva in modo completo.

Per capire meglio ciò, provate a posizionarvi col cursore sull'ultima linea del video e cominciate a digitare delle istruzioni a caso col sistema sopraesposto.

Noterete che, quando il cursore va a capo, l'ultima istruzione della linea precedente rimane spezzata a metà. Siccome, quando si digita un programma, ci si trova sempre sulle ultime linee dello schermo, non era un difetto per niente trascurabile.

Il rimedio è stato trovato intervenendo, con lo stesso sistema, su un'altra routine del sistema operativo, quella che trasforma i comandi BASIC in token e che viene eseguita ogni volta che si preme Return.

I suoi puntatori si trovano nelle locazioni di memoria 772 e 773 e vengono modificati anch'essi dalla Sys 700.

La nuova routine puntata, prima di saltare alla normale gestione dei dati del sistema operativo, controlla se il cursore si trova nelle ultime 3 righe dello schermo e, in caso affermativo, effettua lo scroll del video fino a quando il cursore non giunge alla quartultima riga.

In questo modo, poiché una linea di programma BASIC non può essere più lunga di 4 linee del video, il cursore non raggiungerà mai il fondo dello schermo, a meno che non ci si posizioni volutamente, e il programma funzionerà perfettamente.

Il listato 1, 2 e 3 contengono rispettivamente il programma BASIC, e i disassemblati delle routine di controllo tastiera e della posizione del cursore su video. ■

PERSONAL SOFTWARE

Super Assembler

Utilizziamo il linguaggio macchina con il Commodore 64

di *Gianluca Puccio*

Per iniziare

Il primo programma contiene, sotto forma di linee Data, delle routine in linguaggio macchina che saranno usate dal programma principale.
Dato il Run, le routine sono poste in memoria con delle Poke dopodiché il programma si autodistrugge, quindi è utile registrarlo prima di eseguirlo.
Un controllo finale (checksum) segnala eventuali errori nei Data.
In seguito, ricordarsi sempre di caricare il programma con i Data prima di Super Assembler.

Edit

Dopo il Run appare il menu con 9 opzioni possibili.
Premendo 1 si va in fase di Edit per modificare il programma Assembly presente. Con Shift-1 invece si va in Edit cancellando il programma per cominciare ad inserirne uno nuovo.
In Edit è sempre presente sulla sinistra il simbolo >, che segnala la riga nella quale ci si trova.

Il programma Assembly va introdotto tenendo conto di alcune regole riguardanti la sua struttura.
Ogni linea di programma è suddivisa in tre zone: campo operazione, campo indirizzo e campo etichetta.
La prima linea del programma vuole necessariamente la parola Prog nel campo etichetta.
Prima del programma vero e proprio però possono comparire una o più definizioni di Macro, si vedrà dopo cosa sono. Per ora basta sapere che una definizione di Macro è un

Listato 1. *Il programma principale.*

```
10 REM **********************************
15 REM *                                *
20 REM *   SUPER-ASSEMBLER PER C.64     *
25 REM * ---------------------------- *
30 REM *        GIANLUCA  PUCCIO        *
35 REM *                                *
40 REM **********************************
50 XI=75:XL=300:MX=25:MP=4:EX=INT(XL/10):
K$(0)="<":K$(1)=">":LA=29
60 GOSUB10000
62 DEFFNAZ(X)=C$>="A"ANDC$<="Z"
64 DEFFNNUM(X)=C$>="0"ANDC$<="9"
66 DEFFNALFA(X)=FNAZ(X)ORFNNUM(X)
68 DEFFNESA(X)=FNNUM(X)ORC$>="A"ANDC$<="F
"
70 DIMA$(XL,2),A%(XL),I%(XL),P(XL+1),I$(X
I),ER$(30)
74 K=XI-56
75 DIMMP%(K),MP$(K,MP),MI%(K),MF%(K),PA$(
MP),ET$(EX),ET%(EX)
80 GOSUB20000
90 SP$="
    ":L=0:NL=0:RU=-1
95 REM **********************************
97 REM              M E N U '
99 REM **********************************
100 MO=0:PRINTCHR$(147):POKE53280,9:POKE5
3281,9:POKE198,0
110 PRINTTAB(8)CHR$(144)"1  :  "CHR$(5)"E
DIT"
120 PRINTTAB(88)CHR$(144)"2  :  "CHR$(5)"
ASSEMBLA + SAVE P.SORG."
130 PRINTTAB(88)CHR$(5)"3  :  "CHR$(144)"
SAVE PROGRAMMA SORGENTE"
140 PRINTTAB(48)CHR$(5)"4  :  "CHR$(144)"
LOAD PROGRAMMA SORGENTE"
150 PRINTTAB(48)CHR$(5)"5  :  "CHR$(144)"
SAVE CODICE OGGETTO"
160 PRINTTAB(48)CHR$(5)"6  :  "CHR$(144)"
LOAD CODICE OGGETTO"
170 PRINTTAB(88)CHR$(144)"7  :  "CHR$(5)"
RUN PROGRAMMA"
180 PRINTTAB(88)CHR$(144)"8  :  "CHR$(5)"
DISASSEMBLA"
190 PRINTTAB(88)CHR$(144)"9  :  "CHR$(5)"
MEMORY"
200 GETA$:IFA$=""THEN200
210 A=VAL(A$):PRINTCHR$(147);
220 IFA$="!"THENL=0:NL=0:A=1
230 IFA=1THENMO=1:POKE53280,11:POKE53281,
11:GOTO1000
240 ONAGOTO,3000,7000,7100,8000,8100,8500
,9000,9500
250 :
300 GOTO100
495 REM **********************************
497 REM              ERRORI
499 REM **********************************
500 IFMO<>1THEN600
510 B$(1)="":B$(2)="":OPEN1,3:FORK=1TO2
520 FORI=1TO39:GET#1,C$:B$(K)=B$(K)+C$
530 NEXT:PRINT:NEXT:CLOSE1:PRINTCHR$(145)
```

Seguito listato 1.

```
CHR$(145);
540 PRINT" "SP$:PRINT" "SP$:PRINTCHR$(145
)CHR$(145);
550 PRINTCHR$(156)TAB(ER%)"^":PRINT" "ER$
(ER)
560 POKE198,0:WAIT198,1:POKE198,0:PRINTCH
R$(145)CHR$(145)CHR$(155)B$(1)
570 PRINTB$(2):PRINTCHR$(145)CHR$(145)CHR
$(145)CHR$(29);:ER=0:GOTO1080
590 :
600 PRINTTAB(41)CHR$(156)ER$(ER)CHR$(5):E
R=0:IFMO=0THENGOSUB7630:GOTO100
610 :
620 PRINTTAB(82)"PREMI UN TASTO PER ANDAR
E IN EDIT":WAIT198,1:IFMO=2THEN680
630 A$=AA$:GOSUB7110:IFERTHENER=0:GOTO100
680 L=L-1:MO=1:POKE53280,11:POKE53281,11:
PRINTCHR$(147);:GOTO1000
690 :
700 K=L:L=NL:NEXTL:L=K:GOTO500
995 REM ********************************
997 REM                 EDIT
999 REM ********************************
1000 GOSUB1400
1005 PRINTCHR$(153)">"CHR$(155);
1010 POKE198,0:WAIT198,1:SH=PEEK(653):P=P
EEK(197)
1020 IFP=57THENMO=0:GOTO100
1030 IFP=7THENGETA$:FL=(A$=CHR$(17)ANDL<N
L):GOTO1110
1040 IFP=33ANDSHTHENGETA$:GOTO1170
1050 IFP=18ANDSHTHENGETA$:GOTO1230
1060 C$=CHR$(PEEK(631)):IFNOTFNAZ(X)ANDC$
<>"."ANDC$<>"$"THEN1010
1065 :
1070 L=L+1:IFL>XLTHENER=8:ER%=0:GOTO500
1075 IFL>NLTHENNL=L
1080 PRINTSP$:PRINTCHR$(145)CHR$(29);:GOS
UB60000:GOSUB1500:IFERTHEN500
1090 GOSUB1800:IFERTHEN500
1095 GOSUB2000:IFERTHEN500
1097 PRINTCHR$(145)" ":GOTO1005
1100 :
1110 B$=CHR$(157)+" "+A$(L+1,0)+" "+A$(L+
1,1)
1115 IFFLTHENL=L+1:PRINTB$;:POKE212,0:PRI
NTTAB(LA)":"A$(L,2):GOTO1005
1120 IFA$=CHR$(145)ANDL>0THEN1140
1130 GOTO1010
1140 L=L-1:IFPEEK(214)THENPRINTCHR$(157)"
 "CHR$(157)CHR$(145);:GOTO1005
1150 SYS40779:PRINTSP$:I=L+1:PRINT" "CHR$
(145)CHR$(157)" "A$(I,0)" "A$(I,1);
1160 POKE212,0:PRINTTAB(LA)":"A$(I,2):PRI
NTCHR$(145);:GOTO1005
1170 IFL=NLTHEN1010
1175 L=L+1:NL=NL+1
1180 FORI=NLTOL+1STEP-1:FORJ=0TO2:A$(I,J)
=A$(I-1,J):NEXTJ,I
1185 IFPEEK(214)<24THENSYS40779
1190 PRINTSP$:PRINT" "CHR$(145);
1200 GOSUB60000:PRINTCHR$(145)" ":GOTO100
5
1230 IFL=NLTHEN1010
1240 NL=NL-1
1250 IFL=NLANDL=0THENPRINTSP$:PRINTCHR$(1
45);:GOTO1005
```

gruppo di linee, la prima delle quali vuole la parola Macro nel campo etichetta.

Adesso ci vuole un esempio per chiarire un po' le cose:

```
.★=  828   :PROG
DEC  $FB   :LOOP
BNE  LOOP
RTS
```

Dopo aver digitato Shift-1 da menu, introdurre ".★=" (tra un momento si vedrà cosa significa). Dopo il terzo carattere, il cursore (finto) salta uno spazio e si posiziona all'inizio del campo indirizzo, dove bisogna battere 828. A questo punto occorre inserire un'etichetta (Prog); per farlo basta premere il tasto dei due punti (:), ed il cursore salterà sulla destra dello schermo, nel campo etichetta. Digitare Prog e Return.

Se non è stato commesso alcun errore la linea viene accettata, ed il simbolo > di inizio riga scende di un posto. Si prosegue quindi come appena visto fino all'ultima istruzione. Non tutti i caratteri possono essere visualizzati in una linea; hanno effetto solamente quelli il cui simbolo compare sulla parte superiore del tasto.

E tra questi sono esclusi quelli che svolgono una funzione, come Clr, Crsr, ecc. Fanno eccezione i tasti Del e Return, che agiscono come al solito.

Come si vede dall'esempio, i numeri possono essere espressi in notazione decimale o esadecimale, in questo caso preceduti dal simbolo $.

Se si commette qualche errore nel digitare una linea, appare un messaggio esplicativo sul tipo di errore commesso e l'indicazione della posizione dello stesso nella linea.

Premendo un tasto qualsiasi il messaggio sparisce, la linea viene cancellata ed il cursore è posto al suo inizio, pronto per la correzione. Quando si preme Return per memorizzare una linea, il cursore sparisce e l'unica indicazione della nostra posizione sullo schermo è data dal

simbolo >. In questa situazione (cioè con il cursore assente) sono possibili varie funzioni. Intanto sono abilitati i comandi di Crsr Up e Crsr Down che permettono di posizionarsi su una diversa riga di schermo.

Shift-D elimina la linea alla quale ci si trova.

Shift-I permette l'inserimento di una linea tra quella che precede e quella alla quale ci si trova.

Infine con la freccia a sinistra (←) si torna al menu.

Questi comandi hanno reso necessario l'uso di routine in linguaggio macchina, che implementano lo scroll verticale in entrambe le direzioni, e partendo da una linea qualsiasi (vedi cancellazione ed inserimento linee).

Direttive

Una direttiva è un comando dato all'assemblatore, e va inserita nel programma come se fosse un'istruzione.

Nell'esempio di programma visto prima era presente la direttiva "★=". Con essa si dice all'assemblatore dove dovrà essere memorizzato il programma; nell'esempio: da 828 in poi. Questa direttiva, se presente, deve trovarsi alla prima linea del programma. Nel caso non compaia, l'indirizzo di partenza è fissato a 49152 ($C000 in esadecimale).

La seconda direttiva è .AD (address) e permette di assegnare un'etichetta ad un indirizzo in memoria. Dichiarando, per esempio, all'inizio:

.AD $FFE4 :GET

Si può digitare GET al posto di $FFE4 ogni volta che se ne ha bisogno.

L'indirizzo della .AD può anche essere una locazione in Zero Page.

La terza direttiva è .GO e indica il punto da cui far partire l'esecuzione mediante Run Programma da me-

Seguito listato 1.

```
1260 IFL<>NLTHEN1270
1262 IFPEEK(214)THENPRINTCHR$(157)SP$CHR$
(145):PRINTCHR$(145);:L=L-1:GOTO1005
1264 PRINTSP$:PRINTCHR$(145);:GOTO1005
1270 FORI=L+1TONL:FORJ=0TO2:A$(I,J)=A$(I+
1,J):NEXTJ,I
1280 IFPEEK(214)=24THENPRINTCHR$(157)SP$C
HR$(145):PRINTCHR$(145);:L=L-1:GOTO1005
1285 SYS40704:K=24-PEEK(214):FORI=1TOK:PR
INTTAB(40);:NEXT:PRINTCHR$(157)SP$;
1290 FORI=1TOK:PRINTCHR$(145);:NEXT:PRINT
:PRINTCHR$(145);:POKE241,135:GOTO1005
1395 REM ********************************
1397 REM        PRESENTAZIONE  EDIT
1399 REM ********************************
1400 PRINTCHR$(155);:IFL=0THEN1420
1405 K=L-10:IFK<=0THENK=1
1410 FORI=KTOL:PRINT" "A$(I,0)" "A$(I,1);
:POKE212,0:PRINTTAB(LA)":"A$(I,2):NEXT
1420 I=0
1430 IFL>=NLORI>12THEN1460
1440 I=I+1:L=L+1:PRINT" "A$(L,0)" "A$(L,1
);:POKE212,0
1450 PRINTTAB(LA)":"A$(L,2):GOTO1430
1460 IFITHENFORJ=1TOI:L=L-1:PRINTCHR$(145
);:NEXT
1470 RETURN
1495 REM ********************************
1497 REM    VERIFICA OPERANDO (EDIT)
1499 REM ********************************
1500 C$=LEFT$(A$(L,0),1)
1510 IFFNAZ(X)THEN1700
1520 IFC$="."THEN1570
1530 IFC$<>"$"THENER=1:ER%=1:RETURN
1540 FORI=2TO3:C$=MID$(A$(L,0),I,1)
1550 IFNOTFNESA(X)THENER=1:ER%=I:I=3:NEXT
:RETURN
1560 NEXT:RETURN
1565 :
1570 A$=MID$(A$(L,0),2)
1580 IFA$="*="THEN1630
1590 IFA$="AD"ORA$="AR"THEN1620
1600 IFA$="GO"ORA$="NL"THENRETURN
1610 ER=2:ER%=2:RETURN
1620 IFA$(L,2)=""THENER=17:ER%=LA+1:RETUR
N
1630 A$=A$(L,1):IFA$=""THENER=16:ER%=5:RE
TURN
1640 I=1:C$=LEFT$(A$,1):GOSUB2510
1650 IFC$<>""THENER=1:ER%=4+I
1660 RETURN
1690 :
1700 A$=MID$(A$(L,0),2)
1710 C$=LEFT$(A$,1):IFNOTFNAZ(X)THENER=1:
ER%=2:RETURN
1720 C$=MID$(A$,2):IFNOTFNAZ(X)THENER=1:E
R%=3
1730 RETURN
1795 REM ********************************
1797 REM    VERIFICA ETICHETTA (EDIT)
1799 REM ********************************
1800 A$=A$(L,2):IFA$="MACRO"THEN1900
1810 IFA$=""THENRETURN
1815 C$=LEFT$(A$,1)
1820 IFNOTFNAZ(X)THENER=1:ER%=LA+1:RETURN
1830 A$=MID$(A$,2):IFA$=""THENRETURN
1840 FORI=1TOLEN(A$):C$=MID$(A$,I,1)
1850 IFNOTFNALFA(X)THENER=1:ER%=LA+1+I:I=
LEN(A$)
1860 NEXT:RETURN
1890 :
1900 A$=A$(L,1):I=0:K=0:C$=LEFT$(A$,1)
1910 IFC$=""THENRETURN
1920 I=I+1:K=K+1:IFK>MPTHENER=10:ER%=4+I:
```

Seguito listato 1.

```
RETURN
1930 C$=MID$(A$,I,1):IFNOTFNAZ(X)THEN1970
1940 I=I+1:C$=MID$(A$,I,1)
1950 IFFNALFA(X)THEN1940
1960 IFC$=";"ORC$=""THEN1910
1970 ER=1:ER%=4+I:RETURN
1995 REM ******************************
1997 REM    VERIFICA INDIRIZZO (EDIT)
1999 REM ******************************
2000 IFA$(L,2)="MACRO"THENRETURN
2005 I=0:P=0:A$=A$(L,1):IFA$=""THENRETURN
2010 I=I+1:P=P+1:IFP>MPTHENER=10:ER%=4+I:
RETURN
2020 IFP<>2THEN2050
2030 FORK=0TO55:IFA$(L,0)=I$(K)THENK=55:N
EXT:I=I-1:GOTO2300
2040 NEXT
2050 C$=MID$(A$,I,1):IFC$<>"#"THEN2120
2060 I=I+1:C$=MID$(A$,I,1)
2070 IFC$="<"ORC$=">"THENI=I+1:C$=MID$(A$
,I,1):GOTO2100
2080 GOSUB2500:IFERTHENRETURN
2090 GOTO2280
2100 GOSUB2700:IFERTHENRETURN
2110 GOTO2280
2120 IFC$="'"THENI=I+2:GOTO2270
2130 IFC$<>"("THEN2200
2140 I=I+1:C$=MID$(A$,I,1)
2150 GOSUB2500:IFERTHENRETURN
2160 IFMID$(A$,I,3)=",X)"THENI=I+3:GOTO22
70
2170 IFC$<>")"THEN2300
2180 I=I+1:IFMID$(A$,I,2)=",Y"THENI=I+2
2190 GOTO2270
2200 IFC$<>"<"ANDC$<>">"THEN2240
2210 I=I+1:C$=MID$(A$,I,1)
2220 GOSUB2520:IFERTHENRETURN
2230 GOTO2280
2240 GOSUB2500:IFERTHENRETURN
2250 C$=MID$(A$,I,2)
2260 IFC$=",X"ORC$=",Y"THENI=I+2
2270 C$=MID$(A$,I,1)
2280 IFC$=""THENRETURN
2290 IFC$=";"THEN2010
2300 ER=1:ER%=4+I:RETURN
2497 REM ******************************
2498 REM       VERIFICA DATO  (EDIT)
2499 REM ******************************
2500 IFFNAZ(X)THEN2580
2510 IFC$="$"THENI=I+1:C$=MID$(A$,I,1):GO
TO2550
2520 K=I:IFNOTFNNUM(X)THEN2610
2530 I=I+1:C$=MID$(A$,I,1):IFFNNUM(X)THEN
2530
2535 IFVAL(MID$(A$,K))>65535THEN2620
2540 RETURN
2550 K=I:IFNOTFNESA(X)THEN2610
2560 I=I+1:C$=MID$(A$,I,1):IFFNESA(X)THEN
2560
2565 IFLEN(MID$(A$,K,I-K))>4THEN2620
2570 RETURN
2580 I=I+1:C$=MID$(A$,I,1):IFFNALFA(X)THE
N2580
2590 IFC$="+"ORC$="-"THENI=I+1:C$=MID$(A$
,I,1):GOTO2520
2600 RETURN
2610 ER=1:ER%=4+I:RETURN
2620 ER=18:ER%=4+K:RETURN
2697 REM ******************************
2698 REM       VERIFICA LABEL (EDIT)
2699 REM ******************************
2700 IFNOTFNAZ(X)THENER=1:ER%=4+I:RETURN
2710 I=I+1:C$=MID$(A$,I,1):IFFNALFA(X)THE
N2710
2720 RETURN
```

nu. Se assente, quando scegliamo l'opzione Run sarà richiesta una locazione di partenza.

La quarta direttiva è .AR (array) ed è l'analogo di una Dim del BASIC. Esempio:

```
.AR    50      :VET
```

definisce un vettore di 50 elementi, richiamabili usando Vet, Vet+1, ... Vet+49 oppure utilizzando i registri X e Y, esempio Sta Vet,Y.

La quinta direttiva è .NL ed è una direttiva nulla. La sua utilità è legata soprattutto alle Macro (vedi).

Infine un'ulteriore possibilità, quella di inserire dei dati all'interno del programma, analogamente ad un'istruzione Data del BASIC. È sufficiente introdurre il valore esadecimale, ovviamente preceduto da $, nel campo operazione della linea. Esempio:

```
LDA   LOC    :PROG
STA   $400
LDA   LOC+1
STA   $D800
RTS
$40          :LOC
$07
```

Particolarità del linguaggio

Ci sono alcuni aspetti formali del linguaggio che non sono ancora stati visti.

È permessa una forma particolare dell'indirizzamento immediato: un carattere preceduto dall'apice (') viene interpretato come il codice ASCII corrispondente. Quindi scrivere Lda 'A è lo stesso che scrivere Lda # 65.

Per quanto riguarda l'uso delle etichette, è possibile fare riferimento alla sola parte alta o a quella bassa dell'indirizzo corrispondente. La sintassi è, rispettivamente, > Label e < Label.

Per esempio, se è stata assegnata l'etichetta Get all'indirizzo $FFE4, scrivere > Get oppure $FF è uguale, e lo stesso vale per <Get o $E4.

Ci sono casi nei quali questa possibilità è quasi indispensabile, per esempio quando ci si deve riferire all'in-

dirizzo di un'etichetta che sarà definita più avanti.
Infine la possibilità di specificare quante linee saltare (avanti o indietro) in un'istruzione di Branch. Il numero di linee da saltare, preceduto da < se indietro e da > se avanti, va inserito nel campo indirizzo al posto della usuale etichetta. L'istruzione Beq >5 salta 5 linee avanti se la condizione è verificata, mentre Bcc < 2 salta 2 linee indietro se la Bcc ha esito vero.
L'assemblatore stesso usa questa sintassi durante la fase di espansione Macro, trasformando i salti per etichetta in salti relativi.

Parametri

Prima di iniziare a parlare di Macro, bisogna affrontare un altro discorso: quello dei *parametri formali* e dei *parametri effettivi*.
In BASIC esiste un'istruzione, la Def FN, che permette all'utente di definire una funzione. Se, per esempio, si vuole una funzione che raddoppi un numero dato, si scriverà:

100 DEFFND(X)=2★X

È importante notare che la X usata in questa definizione non ha niente a che fare con la variabile X, la quale può comparire in qualsiasi altra linea per altri scopi. La X dell'esempio è un *parametro formale* che descrive come sarà trasformato il valore introdotto. La linea:

130 X=35:H=FND(6):PRINTX

stamperà 35, lasciando inalterato il valore di X anche dopo la chiamata della funzione Fnd. Il valore di H sarà 12 perché 6 è il *parametro effettivo* che viene passato alla funzione, che svolge 2★6 (=12) e non 2★X.

Macro

Scrivendo un programma Assembly, capita di dover inserire va-

Seguito listato 1.

```
2995 REM ********************************
2997 REM            ASSEMBLA   (MACRO)
2999 REM ********************************
3000 IFNL=0THENPRINTCHR$(147):ER=6:GOTO50
0
3003 AA$="":PRINTCHR$(147)CHR$(5);
3005 PRINTTAB(82);:INPUT"NOME DEL PROGRAM
MA ";AA$:IFLEN(AA$)>15THEN3003
3007 IFAA$=""THEN100
3010 MO=2:IT=55:L=1:PRINTCHR$(144)TAB(120
)"ESAME MACRO"
3020 IFA$(L,2)="MACRO"THEN3050
3030 IFA$(L,2)="PROG"THEN5000
3040 ER=4:GOTO500
3050 IT=IT+1:IFIT>XITHENER=5:GOTO500
3052 FORI=0TOIT-1:IFA$(L,0)=I$(I)THENER=9
3054 NEXT:IFERTHEN500
3060 I$(IT)=A$(L,0)
3070 M=IT-56:MP%(M)=0
3080 REM  MEM.NOME PAR.FORM.E LORO NUM.
3090 A$=A$(L,1):K=0
3100 K=K+1:I=0:IFA$=""THEN3150
3110 I=I+1:C$=MID$(A$,I,1)
3120 IFC$<>";"ANDC$<>""THEN3110
3130 MP$(M,K)=LEFT$(A$,I-1)
3140 A$=MID$(A$,I+1):GOTO3100
3150 MP%(M)=K-1
3160 ML%=0:MI%(M)=L+1
3170 REM                         CICLO 2
3180 L=L+1:ML%=ML%+1
3190 IFL>NLTHENER=6:GOTO500
3200 FL=0
3210 IFA$(L,2)="MACRO"ORA$(L,2)="PROG"THE
N3800
3220 IFA$(L,0)=".NL"THENFL=A$(L,2)<>"":GO
TO3800
3230 IFML%>MXTHENER=13:GOTO500
3240 A$=A$(L,0):C$=LEFT$(A$,1)
3250 IFNOTFNAZ(X)THENER=11:GOTO500
3260 FL=0:FORI=0TOIT-1
3270 IFA$=I$(I)THENFL=1:K=I:I=IT
3280 NEXT:IFFL=0THENER=3:GOTO500
3290 IFK<56THENA%(L)=K:GOTO3180
3294 GOSUB3300:IFERTHEN500
3296 L=L+K:NL=NL+K:ML%=ML%+K:GOTO3180
3298 REM                ESPANSIONE MACRO
3300 MC=K-56
3310 IFNL+MF%(MC)-MI%(MC)>XLTHENER=8:RETU
RN
3320 REM  MEM.NOME PAR.ATT. E LORO NUM.
3330 A$=A$(L,1):K=0
3340 K=K+1:I=0:IFA$=""THEN3390
3350 I=I+1:C$=MID$(A$,I,1)
3360 IFC$<>";"ANDC$<>""THEN3350
3370 PA$(K)=LEFT$(A$,I-1)
3380 A$=MID$(A$,I+1):GOTO3340
3390 IFK-1<>MP%(MC)THENER=12:RETURN
3400 :
3410 K=MF%(MC)-MI%(MC)
3420 IFKTHENFORI=NLTOL+1STEP-1:A%(I+K)=A
(I):FORJ=0TO2:A$(I+K,J)=A$(I,J):NEXTJ,I
3430 :
3440 FORI=LTOL+K:ML=MI%(MC)+I-L
3450 A$(I,0)=A$(ML,0):A$(I,1)=A$(ML,1):IF
I>LTHENA$(I,2)=""
3460 A%(I)=A%(ML)
3470 IFMP%(MC)=0THEN3510
3480 FORW=1TOMP%(MC)
3490 IFA$(ML,1)=MP$(MC,W)THENA$(I,1)=PA$(
W)
3500 NEXTW
3510 NEXTI
3520 RETURN
3790 :
3800 MF%(M)=MI%(M)+ML%-2:REM USCITA C.2
3810 IFML%<2THENER=7:GOTO500
3820 FORI=MI%(M)TOMF%(M)
3830 IFA$(I,1)=""THEN3870
3840 FORW=MI%(M)TOMF%(M)-FL
3850 IFA$(I,1)=A$(W,2)THENA$(I,1)=K$((W<I
```

Seguito listato 1.

```
)+1)+MID$(STR$(I-W),2):W=MF%(M)-FL
3860 NEXTW
3870 NEXTI
3880 FORW=MI%(M)TOMF%(M)-FL:A$(W,2)="":NE
XT
3890 L=L-(A$(L,0)=".NL"ANDA$(L,2)<>"PROG"
):GOTO3020
3995 REM ********************************
3997 REM        CONV. ESA>DEC  (IN A$)
3999 REM ********************************
4000 J=1:IFZ$<>"$"THENJ=2
4010 J=J+1:C$=MID$(A$,J,1)
4020 IFNOTFNESA(X)THENER=1:RETURN
4030 DE=0
4040 DE=16*DE+ASC(C$)-48+7*(C$>"9")
4050 J=J+1:C$=MID$(A$,J,1)
4060 IFFNESA(X)THEN4040
4070 A$=MID$(STR$(DE),2)+MID$(A$,J)
4080 IFZ$<>"$"THENA$=Z$+A$
4090 RETURN
4495 REM ********************************
4497 REM        CONVERSIONE DEC>ESA
4499 REM ********************************
4500 ES$=""
4510 FORW=1TO2:C=DE-INT(DE/16)*16:ES$=CHR
$(C+48-7*(C>9))+ES$
4520 DE=INT(DE/16):NEXT:RETURN
4995 REM ********************************
4997 REM           ASSEMBLA  (PROG)
4999 REM ********************************
5000 IP=L:EN=0:ET$(1)="":PRINT"ESPANSIONE
 MACRO NEL PROGRAMMA"
5010 A$=A$(L,0):C$=LEFT$(A$,1)
5020 IFFNAZ(X)THEN5060
5030 B$=RIGHT$(A$,2)
5040 A%(L)=(B$="*=")+2*(B$="AD")+3*(B$="A
R")+4*(B$="GO")+5*(B$="NL")+7*(C$="$")
5050 GOTO5120
5060 FL=0:FORI=0TOIT
5070 IFA$=I$(I)THENFL=1:K=I:I=IT
5080 NEXT:IFFL=0THENER=3:GOTO500
5090 IFK<56THENA%(L)=K:GOTO5120
5100 GOSUB3300:IFERTHEN500
5110 L=L+K:NL=NL+K
5120 L=L+1:IFL<=NLTHEN5010
5125 MO=3:A$=AA$:PRINTCHR$(155)"REGISTRO
: "A$
5130 GOSUB7010:IFERTHENER=0:GOTO100
5140 PRINTCHR$(144)"CONVERS. ESA>DEC E ME
MORIZZAZ. LABEL"
5150 FORL=IPTONL
5160 A$=A$(L,1):Z$=LEFT$(A$,1)
5170 IFZ$<>"$"ANDMID$(A$,2,1)<>"$"THEN520
0
5180 GOSUB4000:IFERTHEN700
5190 A$(L,1)=A$
5200 A$=A$(L,2):REM          MEM. LABEL
5210 IFA$=""THEN5260
5220 FORJ=1TOEN:IFA$=ET$(J)THENER=14
5230 NEXT:IFERTHEN700
5240 EN=EN+1:IFEN>EXTHENER=15:GOTO700
5250 ET$(EN)=A$:ET%(EN)=L
5260 NEXTL
5265 :
5270 PRINT"TROVA TIPO INDIRIZZAMENTO"
5280 P(IP)=49152:RU=-1
5290 FORL=IPTONL:A$=A$(L,0):B$=A$(L,1)
5300 IFLEFT$(A$,1)<>"."THEN5350
5310 IFA$=".*="THENP(IP)=VAL(A$(L,1)):P(L
+1)=P(IP):GOTO5720
5320 IFA$=".AD"ORA$=".NL"THENP(L+1)=P(L):
GOTO5720
5330 IFA$=".AR"THENP(L+1)=P(L)+VAL(B$):GO
TO5720
5340 IFA$=".GO"THENRU=P(L):P(L+1)=RU:GOTO
5720
5350 IFB$=""THENI%(L)=0:P(L+1)=P(L)+1:GOT
O5720
5360 I=1:C$=LEFT$(B$,1):IFC$="("THENI=2
5370 IFC$="#"THENI=2:C$=MID$(B$,2,1)
```

rie volte determinate sequenze di istruzioni, quali l'incremento di un byte con riporto nel byte successivo, l'azzeramento di un byte, ecc.
Sarebbe comodo poter definire una sola volta, all'inizio del programma, una certa sequenza, usandola poi nel programma principale semplicemente inserendo il suo nome, eventualmente passando anche dei parametri. Tutto ciò è reso possibile dalle Macro-istruzioni. Come esempio, definiamo una Macro che azzera una cella di memoria.

```
AZZ   X      :MACRO
LDA   #0
STA   X
```

Analogamente al Def FN del BASIC, anche in questo caso la X è solamente un parametro formale, che descrive gli effetti di una successiva chiamata della Macro AZZ.
Volendo azzerare la locazione $FB basterà scrivere AZZ $FB.
Ogni volta che l'assemblatore incontrerà l'istruzione AZZ param. inserirà al posto di questa le due istruzioni Lda #0 e Sta param.
Il parametro effettivo può comparire sotto forma di uno dei possibili modi di indirizzamento compatibili con l'istruzione Sta. Quindi AZZ 46 verrà sostituito con LDA #0 e STA 46. Ma potrebbe essere AZZ $C000,X oppure AZZ (45), Y ecc.
La prima linea di ogni definizione di Macro deve contenere Macro nel campo etichetta. La definizione è completa se nella linea successiva c'è Macro o Prog nel campo etichetta, oppure se inserisce la direttiva .NL come nel seguente esempio:

```
PRT   CAR    :MACRO
LDA   CAR
CMP   'Z
BPL   FINE
JSR   $FFD2
.NL          :FINE
```

La funzione è: confrontare il valore

introdotto con il codice ASCII di Z; se superiore non si fa nulla, altrimenti si stampa il relativo carattere su video.
Per stampare una G basterà digitare: Prt 'G.
La direttiva .NL era necessaria in quanto l'istruzione Bpl può saltare direttamente all'esterno della definizione. Per questo durante la fase di espansione Macro l'assemblatore sostituirà Bpl Fine con Bpl >2.
Definiamo una Macro che incrementa l'accumulatore, una istruzione che il 6510 non ha:

```
INA          :MACRO
CLC
ADC    #1
```

Non è presente alcun parametro, quindi la chiamata Ina sarà semplicemente sostituita dalle due istruzioni definite.
La prossima Macro scambia i valori tra due locazioni:

```
SCM    M;N    :MACRO
LDA    M
PHA
LDA    N
STA    M
PLA
STA    N
```

È possibile che all'interno di una Macro se ne usi un'altra, sempre che quest'ultima sia già stata definita. L'ordinamento in modo crescente di due celle di memoria mostra proprio questo:

```
ORD    I;J      :MACRO
LDA    J
CMP    I
BPL    EXIT
SCM    I;J
.NL             :EXIT
```

Ecco ora un programma che scambia tra loro la pagina video e quella del colore, sfruttando la Macro SCM:

```
LDY    #0                 :PROG
SCM    $400,Y;$D800,Y     :LOOP
```

Seguito listato 1.

```
5380 IFC$="<"ORC$=">"THENI=I+1
5390 C$=MID$(B$,I,1):IFFNNUM(X)ORC$="'"TH
ENID=VAL(MID$(B$,I)):GOTO5490
5400 K=I
5410 I=I+1:C$=MID$(B$,I,1):IFFNALFA(X)THE
N5410
5420 Z$=MID$(B$,K,I-K)
5430 FL=0:FORJ=1TOEN:IFZ$=ET$(J)THENFL=J:
J=EN
5440 NEXT:IFFL=0THENER=19:GOTO700
5450 K=ET%(FL):IFA$(K,0)<>".AD"THENID=100
0:GOTO5490
5460 ID=VAL(A$(K,1))
5470 IFC$="+"THENID=ID+VAL(MID$(B$,I+1))
5480 IFC$="-"THENID=ID-VAL(MID$(B$,I-1)
5490 I=1:C$=LEFT$(B$,1):IFC$<>"#"THEN5540
5500 A$(L,1)=MID$(B$,2):I%(L)=1:P(L+1)=P(
L)+2
5510 C$=LEFT$(A$(L,1),1):IFC$="<"ORC$=">"
THEN5720
5520 IFID>255THENER=20:GOTO700
5530 GOTO5720
5540 IFC$<>"'"THEN5570
5550 C$=MID$(B$,2):IFC$=""THENC$=" "
5560 A$(L,1)=MID$(STR$(ASC(C$)),2):I%(L)=
1:P(L+1)=P(L)+2:GOTO5720
5570 IFNOTFNALFA(X)THEN5650
5580 FL=FNAZ(X)ANDB$=Z$ANDLEFT$(A$,1)="B"
ANDA$<>"BIT"ANDA$<>"BRK"
5590 IFFLTHENI%(L)=10:P(L+1)=P(L)+2:GOTO5
720
5600 J=2:P(L+1)=P(L)+2:IFID>255THENJ=5:P(
L+1)=P(L)+3
5610 C$=RIGHT$(B$,2):K=LEN(B$)
5620 IFC$=",X"THENJ=J+1:A$(L,1)=LEFT$(B$,
K-2)
5630 IFC$=",Y"THENJ=J+2:A$(L,1)=LEFT$(B$,
K-2)
5640 I%(L)=J:GOTO5720
5650 IFC$="<"ORC$=">"THENI%(L)=10:P(L+1)=
P(L)+2:GOTO5720
5660 B$=MID$(B$,2):K=LEN(B$):C$=RIGHT$(B$
,3)
5670 IFC$=",X)"THENI%(L)=8:GOTO5700
5680 IFC$="),Y"THENI%(L)=9:GOTO5700
5690 A$(L,1)=LEFT$(B$,K-1):I%(L)=11:P(L+1
)=P(L)+3:GOTO5720
5700 IFID>255THENER=20:GOTO700
5710 A$(L,1)=LEFT$(B$,K-3):P(L+1)=P(L)+2
5720 Z$=LEFT$(A$,1):IFZ$="$"THENGOSUB4000
:GOTO5770
5730 IS=A%(L):IFIS<0THEN5770
5740 GOSUB63500:FORI=0TOI%(L):READA:NEXT
5750 IFA=0ANDA$<>"BRK"THENER=21:GOTO700
5760 A$=MID$(STR$(A),2)
5770 A$(L,0)=LEFT$(A$+"   ",3):NEXTL
5780 PRINT"SOSTITUZ. LABEL CON INDIRIZZO
EFFETTIVO"
5790 FORL=IPTONL:A$=A$(L,1):IFA$=""THEN59
90
5800 I=1:C$=LEFT$(A$,1):B$=""
5810 IFC$="<"ORC$=">"THENB$=C$:A$=MID$(A$
,2):C$=LEFT$(A$,1)
5820 IFFNNUM(X)THENID=VAL(A$):GOTO5940
5830 I=I+1:C$=MID$(A$,I,1):IFFNALFA(X)THE
N5830
5840 Z$=LEFT$(A$,I-1)
5850 FORJ=1TOEN:IFZ$=ET$(J)THENK=ET%(J):J
=EN
5860 NEXT:IFA$(K,0)=".AD"THENID=VAL(A$(K,
1)):GOTO5880
5870 ID=P(K)
5880 IFC$="+"THENID=ID+VAL(MID$(A$,I+1))
5890 IFC$="-"THENID=ID-VAL(MID$(A$,I+1))
5900 IFB$=">"THENID=INT(ID/256):GOTO5980
5910 IFB$="<"THENID=ID-INT(ID/256)*256:GO
TO5980
5920 IFI%(L)<>10THEN5980
5930 GOTO5960
```

Seguito listato 1.

```
5940 IFB$=""THEN5980
5950 K=L+ID*(2*(B$="<")+1):IFK<IPORK>NLTH
ENER=23:GOTO700
5960 ID=P(K)-P(L):IFID<-126ORID>129THENER
=22:GOTO700
5970 ID=(ID+254)AND255
5980 A$(L,1)=STR$(ID)
5990 NEXTL
6000 PRINT"TRASFORMAZIONE FINALE IN CODIC
E OGGETTO"
6010 FORL=IPTONL:IFLEFT$(A$(L,0),1)="."TH
EN6050
6020 A=VAL(A$(L,0)):POKEP(L),A:IFI%(L)=0T
HEN6050
6030 K=VAL(A$(L,1)):C=INT(K/256):B=K-256*
C
6040 POKEP(L)+1,B:POKEP(L)+2,C
6050 NEXTL:L=L-1
6060 PRINTCHR$(155)TAB(42)"OK , COMPILAZI
ONE COMPLETA"
6070 DE=P(IP):GOSUB4500:GOSUB4510
6075 PRINTCHR$(158)TAB(82)"INIZIO CODICE
: $"ES$" ("MID$(STR$(P(IP)),2)")"
6080 DE=P(NL+1)-1:GOSUB4500:GOSUB4510
6085 PRINTTAB(44)"FINE CODICE : $"ES$" ("
MID$(STR$(P(NL+1)-1),2)")"
6090 GOSUB7630:GOTO100
6995 REM ******************************
6997 REM    SAVE  PROGRAMMA SORGENTE
6999 REM ******************************
7000 IFNL=0THENER=6:GOTO500
7002 GOSUB7500:IFA$=""THEN100
7005 PRINTCHR$(155)TAB(82)"REGISTRO : "A$
7010 OPEN15,8,15:OPEN2,8,2,"@:"+A$+",S,W"
:GOSUB7600:IFERTHEN7040
7020 PRINT#2,NL
7030 FORI=1TONL:FORJ=0TO2:PRINT#2,A$(I,J)
:NEXTJ,I
7040 CLOSE2:CLOSE15:IFMOTHENRETURN
7050 ER=0:GOTO100
7095 REM ******************************
7097 REM     LOAD  PROGRAMMA SORGENTE
7099 REM ******************************
7100 GOSUB7500:IFA$=""THEN100
7110 PRINTCHR$(155)TAB(80)"CARICO : "A$:O
PEN15,8,15:OPEN2,8,2,A$+",S,R"
7120 GOSUB7600:IFERTHEN7170
7130 INPUT#2,NL
7140 FORI=1TONL:FORJ=0TO2:A$(I,J)=""
7150 GET#2,Z$:IFZ$<>CHR$(13)THENA$(I,J)=A
$(I,J)+Z$:GOTO7150
7160 NEXTJ,I
7170 CLOSE2:CLOSE15:IFMOTHENRETURN
7180 ER=0:L=0:GOTO100
7490 :
7500 POKE53280,6:POKE53281,6
7505 A$="":PRINTCHR$(147);
7510 PRINTTAB(242);:INPUT"NOME DEL PROGRA
MMA ";A$:IFLEN(A$)>15THEN7505
7520 RETURN
7590 :
7600 INPUT#15,A,B$,C,D
7610 IFA<20THENRETURN
7620 ER=1:PRINTCHR$(155)TAB(123)"ERRORE :
 "B$CHR$(144)
7627 REM ******************************
7628 REM          ATTESA  TASTO
7629 REM ******************************
7630 PRINTCHR$(155)TAB(82)"PREMI UN TASTO
 PER TORNARE AL MENU'"
7640 POKE198,0:WAIT198,1:RETURN
7995 REM ******************************
7997 REM        SAVE  CODICE OGGETTO
7999 REM ******************************
8000 IFNL=0THENER=6:GOTO500
8010 GOSUB7500:IFA$=""THEN100
8020 OPEN15,8,15:OPEN2,8,2,"@:"+A$+"%,S,W
":GOSUB7600:IFERTHENER=0:GOTO8060
8030 PRINTCHR$(155)TAB(122)"REGISTRO : "A
$
8040 A=P(IP):B=P(NL+1)-1:PRINT#2,A:PRINT#
2,B:PRINT#2,RU
```

```
SCM   $500,Y;$D900,Y
SCM   $600,Y;$DA00,Y
SCM   $700,Y:$DB00,Y
INY
BNE   LOOP
RTS
```

Ricordarsi sempre di usare lo stesso numero di parametri dichiarati nella definizione.

Conviene dare alle Macro un nome che ricordi facilmente la loro funzione.

Ricordarsi anche che le definizioni di Macro vanno inserite una dopo l'altra prima del programma principale.

Assembla

Quando da menu si sceglie l'opzione 2, Assembla, per prima cosa viene richiesto il nome del programma, usato in seguito per una Save. Rispondendo con il solo Return si torna al menu, e questo vale per tutte le altre opzioni. Dando invece un nome inizia l'assemblaggio, il quale consiste di ben 5 passate per la trasformazione del programma sorgente in codice oggetto. All'inizio di ogni passata viene visualizzato lo scopo della stessa.

I primi due messaggi però corrispondono ad una passata sola.

Dopo il primo, Esame Macro, viene analizzata la struttura di tutte le definizioni di Macro presenti, e sostituita ogni chiamata con la Macro vera e propria. Con il secondo, Espansione Macro nel programma, inizia la sostituzione delle chiamate di Macro dal programma con le linee che compongono la definizione. In caso di errori trovati, l'assemblatore si arresta, appare un messaggio esplicito sul tipo di errore, e con qualunque tasto si va in Edit, dove il simbolo > indica la linea da correggere.

Terminata la prima passata viene

Seguito listato 1.

```
8050 FORI=ATOB:PRINT#2,PEEK(I):NEXT
8060 CLOSE2:CLOSE15:GOTO100
8095 REM ********************************
8097 REM       LOAD  CODICE OGGETTO
8099 REM ********************************
8100 GOSUB7500:IFA$=""THEN100
8110 OPEN15,8,15:OPEN2,8,2,A$+"%,S,R"
8120 GOSUB7600:IFERTHENER=0:GOTO8150
8130 INPUT#2,A,B,RU:PRINTCHR$(155)TAB(122
)"CARICO : "A$
8140 FORI=ATOB:INPUT#2,P:POKEI,P:NEXT
8150 CLOSE2:CLOSE15:GOTO100
8495 REM ********************************
8497 REM           RUN  PROGRAMMA
8499 REM ********************************
8500 POKE53280,0:POKE53281,0:PRINTCHR$(15
2)
8510 IFRU<0THENA$="":GOTO8530
8520 DE=RU:PRINTTAB(9)CHR$(158)"RUN PROGR
AMMA"TAB(40)CHR$(155):GOTO8580
8530 PRINTTAB(123)"NEL PROGRAMMA NON E' P
RESENTE           LA DIRETTIVA [.GO]"
8540 PRINTTAB(83);:INPUT"LOCAZIONE DI INI
ZIO ";A$:IFA$=""THEN100
8550 C$=LEFT$(A$,1)
8560 IFFNNUM(X)THENSYS(VAL(A$)):GOTO8590
8570 Z$=C$:GOSUB4000:IFERTHEN500
8580 SYS(DE)
8590 PRINTCHR$(158):GOSUB7630:GOTO100
8995 REM ********************************
8997 REM            DISASSEMBLA
8999 REM ********************************
```

```
9000 POKE53280,11:POKE53281,11:GOSUB9300:
IFA$=""THEN100
9002 K$=" D I S A S S E M B L A "
9005 K=1:PRINTCHR$(147)CHR$(144)TAB(5)CHR
$(213);:FORI=1TO28:PRINTCHR$(192);
9008 NEXT:PRINTCHR$(201)
9011 FORI=1TO23:PRINTTAB(3)MID$(K$,I,1)"
"CHR$(221)TAB(34)CHR$(221):NEXT
9014 PRINTTAB(5)CHR$(202);:FORI=1TO28:PRI
NTCHR$(192);:NEXT
9017 PRINTCHR$(203)CHR$(19)
9020 DE=H:GOSUB4500:GOSUB4510:PRINTTAB(6)
CHR$(158)"$"ES$;:P=PEEK(H)
9025 DE=P:GOSUB4500:PRINT"  "CHR$(155)ES$
;
9030 POKE65,P1:POKE66,P2:FORI=0TOP:READA$
:NEXT:I=VAL(MID$(A$,4)):IFI=0THEN9060
9040 A$=LEFT$(A$,3):H=H+1:P=PEEK(H):DE=P:
GOSUB4500:PRINT" "ES$;:B$=ES$
9050 IFI>4ANDI<8ORI=11THENH=H+1:P=PEEK(H)
:DE=P:GOSUB4500:PRINT" "ES$;:B$=ES$+B$
9060 PRINTTAB(23)CHR$(153)A$;
9065 ONIGOTO9080,9090,9100,9110,9090,9100
,9110,9120,9130,9140,9150
9070 PRINT:GOTO9160
9080 PRINT" #$"B$:GOTO9160
9090 PRINT" $"B$:GOTO9160
9100 PRINT" $"B$",X":GOTO9160
9110 PRINT" $"B$",Y":GOTO9160
9120 PRINT" ($"B$",X)":GOTO9160
9130 PRINT" ($"B$"),Y":GOTO9160
9140 DE=H+P+1+256*(P>127):GOSUB4500:GOSUB
4510:PRINT" $"ES$:GOTO9160
9150 PRINT" ($"B$")"
```

registrato il programma. L'assemblaggio trasforma materialmente il programma, per cui in caso di errore non sarebbe più utilizzabile per la correzione. Invece così basta una Load prima di tornare in Edit. Il motivo per cui la Save è effettuata solo dopo la prima passata ha ancora a che fare con un eventuale errore.

Dopo la fase di espansione Macro il programma può risultare molto più lungo, e una linea può non occupare la stessa posizione che aveva all'inizio dell'assemblaggio. Se la Save venisse usata subito e poi fosse trovato un errore in una passata diversa dalla prima, la Load (necessaria per tornare in Edit) caricherebbe il programma originale, nel quale la linea dell'errore potrebbe non essere nella stessa posizione. In tal caso, tornati in Edit, la linea indicata non sarebbe quella voluta.

Se l'assemblatore non trova errori il programma, ormai in linguaggio macchina, viene posto in memoria con delle Poke, e vengono indicati il primo e l'ultimo byte utilizzati.

Le altre opzioni

I quattro comandi di Save e Load permettono la registrazione e il caricamento dei programmi, prima o dopo l'assemblaggio.

Si può usare lo stesso nome per registrare sia il programma sorgente sia il codice oggetto, in quanto prima della registrazione di quest'ultimo viene aggiunto automaticamente un carattere speciale al nome del file.

L'opzione Run è già stata vista parlando delle direttive.

Disassembla mostra ovviamente il disassemblato di una zona di memoria, e Memory mostra il contenuto di determinati byte, in esadecimale e come Chr$.

Si può fermare la visualizzazione con qualsiasi tasto, poi riprendere con Return oppure tornare al menu con un altro qualunque. La visualizzazione si ferma automaticamente quando lo schermo è stato riempito.

Modifiche per l'unità a nastri

Eliminare le linee 7120 e 8120.

Eliminare Close 15 dalle linee: 7040, 7170, 8060 e 8150.

Riscrivere come indicato le seguenti linee:

```
7010 OPEN 2,1,1,A$
7110 OPEN 2,1,0,A$
8020 OPEN 2,1,1,A$+"%"
8110 OPEN 2,1,0,A$+"%"
```

■

Seguito listato 1.

```
9160 K=K+1:H=H+1:IFPEEK(198)=0ANDK<24ANDH
<65536THEN9020
9170 POKE198,0:WAIT198,1:GETA$
9180 IFA$<>CHR$(13)ORH>65535THEN100
9190 IFK<24THEN9020
9200 GOTO9005
9295 REM ******************************
9297 REM     INPUT LOCAZIONE INIZIALE
9299 REM ******************************
9300 A$="":PRINTCHR$(147)CHR$(158);
9305 PRINTTAB(202)"";:INPUT"LOCAZIONE INI
ZIALE ";A$:IFA$=""THENRETURN
9310 C$=LEFT$(A$,1):IFFNNUM(X)THENDE=VAL(
A$):GOTO9330
9320 Z$=C$:GOSUB4000
9330 IFER>0ORDE>65535THENER=0:GOTO9300
9340 H=DE:RETURN
9495 REM ******************************
9497 REM           M E M O R Y
9499 REM ******************************
9500 POKE53280,11:POKE53281,11:GOSUB9300:
IFA$=""THEN100
9510 H=INT(H/8)*8
9515 K=1:PRINTCHR$(147)
9520 DE=H:GOSUB4500:GOSUB4510:PRINTCHR$(1
53)"$"ES$" "CHR$(155);
9530 FORI=0TO7:P(I)=PEEK(H+I):DE=P(I):GOS
UB4500:PRINT" "ES$;:NEXT
9535 PRINT" :"CHR$(158);:FORI=0TO7
9540 IFP(I)<32ORP(I)>127ANDP(I)<160THENPR
INT" ";:GOTO9560
9550 PRINTCHR$(P(I));:POKE212,0
9560 NEXT:H=H+8:K=K+1:IFPEEK(198)=0ANDK<2
4ANDH<65536THEN9520
9570 POKE198,0:WAIT198,1:GETA$
9580 IFA$<>CHR$(13)ORH>65535THEN100
9590 IFK<24THEN9520
9600 GOTO9515
9995 REM ******************************
9997 REM          PRESENTAZIONE
9999 REM ******************************
10000 POKE53280,2:POKE53281,2
10010 C$=" S U P E R - A S S E M B L E R"
10020 PRINTCHR$(147)CHR$(156)TAB(211)"QUE
STO E' ..."TAB(200)CHR$(158)
10030 PRINTTAB(7)"----------------------
------":PRINTCHR$(145)CHR$(145)TAB(7);
10200 PRINTCHR$(155);:RETURN
19995 REM *****************************
19997 REM       INIZIALIZZAZIONE
19999 REM *****************************
20000 GOSUB63500:I$(0)=A$:NP=PC
20005 FORK=1TO55:NP=PEEK(NP)+256*PEEK(NP+
1):FORI=5TO7
20010 I$(K)=I$(K)+CHR$(PEEK(NP+I)):NEXTI
20012 IFK=INT(K/2)*2THENPRINTMID$(C$,K,1)
CHR$(29);
20015 NEXTK
20020 POKE65,PEEK(NP):POKE66,PEEK(NP+1)
20030 READA$:READA$
```

ELENCO DELLE VARIABILI PIU' USATE

Vettori

A$(I,J)	J-esimo campo della I-esima linea.
A%(I)	Tipo istruzione della linea I.
I%(I)	Tipo indirizzamento dell'istruzione della linea I.
P(I)	Locazione per l'istruzione della linea I.
PA$(I)	Nome del parametro effettivo I (in chiamata).
MP$(I,J)	Nome del parametro formale J della Macro I.
MP%(I)	Numero parametri della Macro I.
MI%(I)	Linea dove inizia la Macro I (dopo il nome).
MF%(I)	Linea dove termina la Macro I.
ET$(I)	Nome dell'etichetta I.
ET%(I)	Linea dell'etichetta I.
ER$(I)	Descrizione dell'errore I.

Costanti

XI	Numero massimo di istruzioni (standard + Macro).
XL	Numero massimo di linee.
MX	Numero massimo di linee per una Macro.
MP	Numero massimo di parametri per una Macro.
EX	Numero massimo di etichette.

Variabili principali

L	Linea in cui ci si trova.
NL	Numero linee del programma.
IT	Numero istruzioni (standard + Macro).
IP	Linea dove inizia il programma (dopo eventuali Macro).
M	Macro alla quale ci si trova.
ML%	Lunghezza della Macro in cui ci si trova.
EN	Numero etichette presenti nel programma.
ER	Codice dell'errore.

Funzioni (booleane)

FN AZ	Vera se C$ è una lettera.
FN NUM	Vera se C$ è un carattere numerico.
FN ALFA	Vera se C$ è un carattere alfanumerico.
FN ESA	Vera se C$ è un carattere esadecimale.

Seguito listato 1.

```
20040 P=PEEK(65)+256*PEEK(66)-4
20050 P2=INT(P/256):P1=P-256*P2
20060 ER$(1)="ERRORE DI SINTASSI"
20070 ER$(2)="QUESTA DIRETTIVA NON ESISTE
"
20080 ER$(3)="ISTRUZIONE INESISTENTE"
20090 ER$(4)="LA LINEA D'INIZIO E LA PRIM
A DOPO OGNI  MACRO VOGLIONO 'MACRO' O"
20095 ER$(4)=ER$(4)+" 'PROG'"
20100 ER$(5)="HAI SUPERATO IL MAX NUMERO
ACCETTABILE  DI MACRO"
20110 ER$(6)="NON E' PRESENTE ALCUN PROGR
AMMA"
20120 ER$(7)="UNA MACRO NON PUO' ESSERE V
UOTA"
20130 ER$(8)="HAI SUPERATO IL MAX NUMERO
DI LINEE"
20140 ER$(9)="QUESTA ISTRUZIONE ESISTE GI
A'"
20150 ER$(10)="HAI SUPERATO IL MAX NUM. D
I PARAMETRI"
20160 ER$(11)="DIRETTIVA NON LECITA ALL'I
NTERNO DI UNA MACRO"
20170 ER$(12)="IL NUM. DI PARAMETRI ATTUA
LI E' DIVERSO DAL NUM. DI PARAMETRI"
20180 ER$(12)=ER$(12)+" FORMALI"
20190 ER$(13)="HAI SUPERATO IL MAX NUMERO
 ACCETTABILE  DI LINEE IN UNA MACRO"
20200 ER$(14)="ETICHETTA GIA' DEFINITA"
20210 ER$(15)="HAI SUPERATO IL MASSIMO NU
M. POSSIBILE  DI ETICHETTE"
20220 ER$(16)="QUESTA DIRETTIVA VUOLE UN
INDIRIZZO"
20230 ER$(17)="QUESTA DIRETTIVA VUOLE UN
ETICHETTA"
20240 ER$(18)="HAI SUPERATO 65535 ($FFFF)
"
20250 ER$(19)="USO DI UN'ETICHETTA NON DE
FINITA"
20260 ER$(20)="HAI SUPERATO 255 ($FF)"
20270 ER$(21)="TIPO ILLECITO DI INDIRIZZA
MENTO"
20280 ER$(22)="BRANCH TROPPO GRANDE"
20290 ER$(23)="BRANCH A LINEA INESISTENTE
"
20400 RETURN
24997 REM ******************************
24998 REM       DATI PER 'ASSEMBLA'
24999 REM ******************************
25000 DATAADC,,105,101,117,,109,125,121,9
7,113,,
25001 DATAAND,,41,37,53,,45,61,57,33,49,,
25002 DATAASL,10,,6,22,,14,30,,,,,
25003 DATABCC,,,,,,,,,,,144,
25004 DATABCS,,,,,,,,,,,176,
25005 DATABEQ,,,,,,,,,,,240,
25006 DATABIT,,,36,,,44,,,,,,
25007 DATABMI,,,,,,,,,,,48,
25008 DATABNE,,,,,,,,,,,208,
25009 DATABPL,,,,,,,,,,,16,
25010 DATABRK,0,,,,,,,,,,,
25011 DATABVC,,,,,,,,,,,80,
25012 DATABVS,,,,,,,,,,,112,
25013 DATACLC,24,,,,,,,,,,,
25014 DATACLD,216,,,,,,,,,,,
25015 DATACLI,88,,,,,,,,,,,
25016 DATACLV,184,,,,,,,,,,,
25017 DATACMP,,201,197,213,,205,221,217,1
93,209,,
25018 DATACPX,,224,228,,,236,,,,,,
25019 DATACPY,,192,196,,,204,,,,,,
25020 DATADEC,,,198,214,,206,222,,,,,
25021 DATADEX,202,,,,,,,,,,,
25022 DATADEY,136,,,,,,,,,,,
25023 DATAEOR,,73,69,85,,77,93,89,65,81,,
25024 DATAINC,,,230,246,,238,254,,,,,
25025 DATAINX,232,,,,,,,,,,,
25026 DATAINY,200,,,,,,,,,,,
25027 DATAJMP,,,,,,76,,,,,,108
25028 DATAJSR,,,,,,32,,,,,,
25029 DATALDA,,169,165,181,,173,189,185,1
61,177,,
25030 DATALDX,,162,166,,182,174,,190,,,,
25031 DATALDY,,160,164,180,,172,188,,,,,
25032 DATALSR,74,,70,86,,78,94,,,,,
25033 DATANOP,234,,,,,,,,,,,
25034 DATAORA,,9,5,21,,13,29,25,1,17,,
25035 DATAPHA,72,,,,,,,,,,,
25036 DATAPHP,8,,,,,,,,,,,
25037 DATAPLA,104,,,,,,,,,,,
25038 DATAPLP,40,,,,,,,,,,,
25039 DATAROL,42,,38,54,,46,62,,,,,
25040 DATAROR,106,,102,118,,110,126,,,,,
25041 DATARTI,64,,,,,,,,,,,
25042 DATARTS,96,,,,,,,,,,,
25043 DATASBC,,233,229,245,,237,253,249,2
25,241,,
25044 DATASEC,56,,,,,,,,,,,
25045 DATASED,248,,,,,,,,,,,
25046 DATASEI,120,,,,,,,,,,,
25047 DATASTA,,,133,149,,141,157,153,129,
145,,
25048 DATASTX,,,134,,150,142,,,,,,
25049 DATASTY,,,132,148,,140,,,,,,
25050 DATATAX,170,,,,,,,,,,,
25051 DATATAY,168,,,,,,,,,,,
25052 DATATSX,186,,,,,,,,,,,
25053 DATATXA,138,,,,,,,,,,,
25054 DATATXS,154,,,,,,,,,,,
25055 DATATYA,152,,,,,,,,,,,
25097 REM ******************************
25098 REM      DATI PER 'DISASSEMBLA'
25099 REM ******************************
26000 DATABRK,ORA8,???,???,???,ORA2,ASL2,
???,PHP,ORA1,ASL,???,???,ORA5,ASL5,???
26001 DATABPL10,ORA9,???,???,???,ORA3,ASL
3,???,CLC,ORA7,???,???,???,ORA6,ASL6
26002 DATA???,JSR5,AND8,???,???,BIT2,AND2
,ROL2,???,PLP,AND1,ROL,???,BIT5,AND5
26003 DATAROL5,???,BMI10,AND9,???,???,???
,AND3,ROL3,???,SEC,AND7,???,???,???
26004 DATAAND6,ROL6,???,RTI,EOR8,???,???,
???,EOR2,LSR2,???,PHA,EOR1,LSR,???,JMP5
26005 DATAEOR5,LSR5,???,BVC10,EOR9,???,??
?,???,EOR3,LSR3,???,CLI,EOR7,???,???
26006 DATA???,EOR6,LSR6,???,RTS,ADC8,???,
???,???,ADC2,ROR2,???,PLA,ADC1,ROR,???
26007 DATAJMP11,ADC5,ROR5,???,BVS10,ADC9,
???,???,???,ADC3,ROR3,???,SEI,ADC7,???
26008 DATA???,???,ADC6,ROR6,???,???,STA8,
???,???,STY2,STA2,STX2,???,DEY,???,TXA
26009 DATA???,STY5,STA5,STX5,???,BCC10,ST
A9,???,???,STY3,STA3,STX3,???,TYA,STA7
26010 DATATXS,???,???,STA6,???,???,LDY1,L
DA8,LDX1,???,LDY2,LDA2,LDX2,???,TAY
26011 DATALDA1,TAX,???,LDY5,LDA5,LDX5,???
,BCS10,LDA9,???,???,LDY3,LDA3,LDX3,???
```

Super Assembler

Seguito listato 1.

```
26012 DATACLV,LDA7,TSX,???,LDY6,LDA6,LDX6
,???,CPY1,CMP8,???,???,CPY2,CMP2,DEC2
26013 DATA???,INY,CMP1,DEX,???,CPY5,CMP5,
DEC5,???,BNE10,CMP9,???,???,???,CMP3
26014 DATADEC3,???,CLD,CMP7,???,???,???,C
MP6,DEC6,???,CPX1,SBC8,???,???,CPX2
26015 DATASBC2,INC2,???,INX,SBC1,NOP,???,
CPX5,SBC5,INC5,???,BEQ10,SBC9,???,???
26016 DATA???,SBC3,INC3,???,SED,SBC7,???,
???,???,SBC6,INC6,???
59995 REM ******************************
59997 REM             INPUT LINEA
59999 REM ******************************
60000 FORI=0TO2:A$(L,I)="":NEXT
60010 POKE212,0:PRINTCHR$(18)"."CHR$(157)
CHR$(146);
60020 GETA$:IFA$=""THEN60020
60030 A=ASC(A$):P=POS(0):IFA>95THEN60020
60040 IFA>31THEN60130
60050 IFA=13THENPRINT" ":RETURN
60060 IFA<>20THEN60020
60070 IFP>1ANDP<5THENI=0:GOTO60200
60080 IFP>5ANDP<LATHENI=1:GOTO60200
60090 IFP>LA+1THENI=2:GOTO60200
60100 IFP=5THENPRINTA$;:I=0:GOTO60200
60110 IFP<>LA+1THEN60020
60120 FORI=1TOP-PR:PRINTA$;:NEXT:GOTO6002
0
60130 IFA=58THENPR=P:PRINT" "TAB(LA)A$;:G
OTO60010
60140 IFP<4THENI=0:GOTO60300
60150 IFP>4ANDP<LA-1THENI=1:GOTO60300
60160 IFP>LAANDP<LA+6THENI=2:GOTO60300
60170 IFP=4THENPRINT" ";
60180 GOTO60010
60190 :
60200 A$(L,I)=LEFT$(A$(L,I),LEN(A$(L,I))-
1):PRINTA$;
60210 GOTO60010
60300 A$(L,I)=A$(L,I)+A$:PRINTA$;
60310 IFP=3THENPRINT" ";
60320 GOTO60010
63495 REM ******************************
63497 REM            RESTORE IS
63499 REM ******************************
63500 RESTORE:READA$:PC=PEEK(65)+256*PEEK
(66)-8
63501 ID=PEEK(63)+256*PEEK(64)
63502 LN=ID+IS
63503 LN(1)=INT(LN/256)
63504 LN(2)=LN-LN(1)*256
63505 IFIS=0THENRETURN
63506 FORI=1TOIS
63507 NP=PEEK(PC)+256*PEEK(PC+1):PC=NP
63508 NEXT:PC=PC+4
63509 PC(1)=INT(PC/256)
63510 PC(2)=PC-PC(1)*256
63511 POKE63,LN(2):POKE64,LN(1)
63512 POKE65,PC(2):POKE66,PC(1)
63513 READA$:RETURN
```

Commenti al listato

50 - Definisce delle costanti, ovvero delle variabili che non verranno più modificate dal programma.
60 - Manda alla subroutine di presentazione.
62-75 - Definiscono delle funzioni di controllo sul tipo di carattere (esempio alfanumerico, esadecimale, ecc.) e dimensionano tutti i vettori.
80 - Manda alla subroutine che riempie il vettore dei messaggi di errore e quello delle istruzioni standard del microprocessore.
90 - Completa l'inizializzazione delle variabili.
100-190 - Fanno apparire il menu con tutte le opzioni. La variabile MO (modo) che compare in linea 100 è posta a 0 per indicare il modo menu.
Gli altri valori che può assumere sono:
1=Edit, 2=Assembla Macro, 3=Assembla Prog.
200-300 - Gestiscono la scelta dell'opzione di menu.
500-700 - Visualizzano l'errore commesso, ma in maniera diversa a seconda che ci si trovi in Edit o in un altro modo.
• Se in Edit, il messaggio potrebbe sostituire altre linee presenti sullo schermo, quindi prima di stamparle, le linee 510-530 effettuano un Input da schermo per memorizzare le righe dove comparirà il messaggio in modo da poterle ripristinare subito dopo. La variabile ER% contiene la colonna nella quale è stato commesso l'errore.
• Se in Assembla Prog., dopo la visualizzazione del messaggio viene effettuata una LOAD per riprendere il programma originale e poter andare in Edit per la correzione.
• Negli altri casi appare semplicemente il messaggio d'errore.
1000-1060 - Controllano la scelta di uno dei comandi possibili in Edit.
1070-1097 - Servono per l'introduzione di una linea mediante l'uso della subroutine di Input linea (60000-60320), e poi per verificarne la correttezza.
1110-1130 - Spostano verso il basso l'indicatore di riga a seguito di un Crsr Down.
1140-1160 - Spostano verso l'alto l'indicatore di riga (dopo un Crsr Up).
1170-1200 - Permettono l'inserimento di una linea tra due consecutive. La Sys 40779 esegue uno scroll verso il basso a partire dalla riga nella quale ci si trova.
1230-1290 - Cancellano la linea alla quale ci si trova. La Sys 40704 esegue uno scroll verso l'alto fino alla linea alla quale ci si trova.
1400-1470 - Subroutine chiamata ogni volta che si va in Edit. Visualizza una parte del programma assembler presente in memoria.
1500-1730 - Subroutine per la verifica del campo operazione.
1800-1970 - Subroutine per la verifica del campo etichetta.
2000-2300 - Subroutine per la verifica del campo indirizzo.
2500-2620 - Subroutine per la verifica di un generico dato. Controlla, per esempio, che i caratteri successivi al simbolo "$" siano caratteri esadecimali.
2700-2720 - Subroutine per la verifica di un'etichetta: il primo carattere deve essere alfabetico, gli altri (se ci sono) possono essere alfanumerici.
3000-3070 - Viene richiesto il nome del programma ed inizia l'assemblaggio. Se non ci sono definizioni di Macro si passa subito alla fase successiva, Assembla Prog. (da 5000 in poi). Altrimenti viene subito incrementata la variabile IT (numero di istruzioni) per far sì che l'assemblatore riconosca la Macro come una vera istruzione.
3090-3160 - Inizia l'assemblaggio delle Macro. Viene memorizzato il nome dei parametri formali e il loro numero.
3180-3296 - Ciclo nel quale vengono esaminate tutte le istruzioni di una Macro. Se compare la chiamata di un'altra Macro viene effettuata la sostituzione delle linee.
3300-3520 - Subroutine che esegue la sostituzione di una chiamata di Macro, con le linee della sua definizione.
3800-3890 - Sostituiscono le istruzioni di branch per etichetta con istruzioni di branch relativo. Poi l'esecuzione prosegue alla linea 3020 per esaminare la Macro successiva.
4000-4090 - Subroutine per la conversione di una stringa esadecimale (a$) in un numero (DE).
4500-4520 - Subroutine per la conversione di un numero (DE) in una stringa esadecimale (ES$) di due caratteri. Per ottenere una stringa di quattro caratteri viene chiamata questa subroutine una seconda volta, però a 4510.
5000-5120 - Inizia l'assemblaggio del programma principale. Questa è la fase dell'espansione Macro.
5125-5260 - Viene registrato il programma e poi eseguita la seconda passata: conversione di tutti gli indirizzi esadecimali e memorizzazione delle etichette definite.

5270-5770 - Terza passata: è trovato il tipo di indirizzamento di tutte le istruzioni. Viene anche riempito il vettore P(.) con la locazione nella quale porre ogni istruzione. Il tipo di indirizzamento è trovato analizzando alcuni particolari del campo indirizzo, esempio se questo termina con "),Y" oppure ",X", ecc. Alla fine si sostituisce il campo operazione con il codice dell'istruzione, ottenuto con una chiamata alla subroutine 63500.
5780-5990 - Quarta passata: tutte le etichette usate vengono sostituite con il loro indirizzo effettivo facendo attenzione a sintassi del tipo "Pippo+3" oppure del tipo ">Pippo".
6000-6090 - Quinta passata: trasformazione finale in codice oggetto. Il programma è ora pronto per essere posto in memoria con semplici Poke. Vengono mostrati il primo e l'ultimo byte occupati dal codice.
7000-7050 - Save del programma sorgente. Il primo dato è il numero di linee del programma.
7100-7180 - Load del programma sorgente.
7500-7520 - Subroutine di Input per il nome del programma, la cui lunghezza deve essere compresa tra 1 e 15 caratteri. È usata dalle varie routine di Save e Load.
7600-7620 - Subroutine di interrogazione drive per eventuali errori.
7630-7640 - Subroutine di attesa tasto prima di tornare al menu.
8000-8060 - Save del codice oggetto. I primi tre dati sono,

rispettivamente: il primo byte di codice, l'ultimo byte, e la locazione dalla quale far partire l'esecuzione (mediante Run Programma da menu).
8100-8150 - Load del codice oggetto.
8500-8590 - Permettono l'esecuzione di un codice in memoria.
9000-9200 - Mostrano il disassemblato di qualunque zona di memoria. Le variabili P1 e P2 che compaiono alla linea 9030 sono i valori che, assegnati alle locazioni 65 e 66, permettono di iniziare a leggere i Data a partire dalla linea 26000, saltando così tutti quelli precedenti (utilizzati durante l'assemblaggio). Le locazioni 65 e 66 infatti, contengono l'indirizzo del primo dato che verrà letto con una Read.
9300-9340 - Subroutine per la richiesta di una locazione di partenza, con controllo del dato introdotto. È usata per opzioni quali Disassembla e Memory.
9500-9600 - Mostrano il contenuto della zona di memoria voluta, sia come valore esadecimale sia come Chr$.
10000-10200 - Subroutine di presentazione del programma.
20000-20050 - Prima parte della subroutine di inizializzazione. Riempimento del vettore IS$(.) con i nomi delle istruzioni del microprocessore.
Siccome ognuno di questi nomi si trova per primo in un'istruzione Data, non viene usata la Read. Si sfrutta il puntatore che in memoria precede ogni linea BASIC, per passare alla linea successiva.
Il nome viene ricavato leggendo direttamente dalla memoria i tre valori di ogni linea che interessano, e usando la funzione Chr$ per trasformarli in caratteri. Dopo aver letto tutto il blocco di linee Data in questo modo, si trovano i valori da assegnare a P1 e P2 per la successiva lettura del secondo blocco di linee Data.
20060-20400 - Seconda parte della subroutine di inizializzazione. Viene riempito il vettore ER$(.) dei messaggi di errore.
25000-25055 - Primo blocco di linee Data (usato in Assembla). Il primo dato di ogni linea è di codice mnemonico dell'istruzione, gli altri sono i codici relativi a tutti i tipi di indirizzamento di quell'istruzione.
26000-26016 - Secondo blocco di linee Data (usato in Disassembla). Ci sono tutte le istruzioni relative ai codici da 0 a 255 con una cifra che indica il tipo di indirizzamento. Per molti codici non esiste l'istruzione, in tal caso al posto dell'istruzione compare "???".
60000-60320 - Subroutine che gestisce l'Input controllato. A seconda della posizione del cursore sulla linea, viene riempito il primo, il secondo o il terzo campo del vettore A$(..). La locazione 212 che compare in linea 60010 segnala al computer se ci si trova o no in "quote mode". Viene azzerata per evitare problemi in caso di visualizzazione delle virgolette.
63500-63513 - Subroutine che esegue un Restore ad una qualsiasi linea tra 20000 e 20055, in relazione alla variabile IS. Per esempio, se IS=4 la routine posiziona il puntatore all'oggetto Data (locazioni 65 e 66) alla linea 25004, e pone nella variabile A$ il primo dato della linea.

ELENCO DEI MESSAGGI DI ERRORE

1	Errore di sintassi.
2	Questa direttiva non esiste.
3	Istruzione inesistente.
4	La linea d'inizio e la prima dopo ogni Macro vogliono Macro o Prog.
5	Hai superato il massimo numero accettabile di Macro.
6	Non è presente alcun programma.
7	Una Macro non può essere vuota.
8	Hai superato il massimo numero di linee.
9	Questa istruzione esiste già.
10	Hai superato il massimo numero di parametri.
11	Direttiva non lecita all'interno di una Macro.
12	Il numero di parametri attuali è diverso dal numero di parametri formali.
13	Hai superato il massimo numero accettabile di linee in una Macro.
14	Etichetta già definita.
15	Hai superato il massimo numero possibile di etichette.
16	Questa direttiva vuole un indirizzo.
17	Questa direttiva vuole un'etichetta.
18	Hai superato 65535 ($FFFF).
19	Uso di un'etichetta non definita.
20	Hai superato 255 ($FF).
21	Tipo illecito di indirizzamento.
22	Branch troppo grande.
23	Branch a linea inesistente.

Listato 2. *Qui sono contenute, sotto forma di linee Data, delle routine in linguaggio macchina usate dal programma principale.*

```
?998 POKE52,159:POKE56,159:LM=40704:K=0
999 FORI=0TO150:READA:POKELM+I,A:K=K+A:NE
XT
1000 IFK<>25031THENPRINT"ERRORE NEI DATA"
:END
1001 NEW
1010 DATA169,216,133,251,169,3,133,252,16
2,255,24,169,40,101,251,133,251,144
1020 DATA2,230,252,232,228,214,208,240,16
5,252,133,254,24,165,251,105,40,133
1030 DATA253,144,2,230,254,160,39,177,253
,145,251,136,16,249,165,254,133,252
1040 DATA165,253,133,251,24,105,40,133,25
3,144,2,230,254,232,224,24,48,225,24
1050 DATA144,49,169,152,133,251,169,192,1
33,253,169,7,133,252,133,254,56,169
1060 DATA24,229,214,170,160,39,177,251,14
5,253,136,16,249,165,252,133,254,165
1070 DATA251,133,253,56,233,40,133,251,17
6,2,198,252,202,208,227,162,4,169,0
1080 DATA133,251,169,216,133,252,169,15,1
60,255,145,251,136,208,251,145,251,230
1090 DATA252,202,208,242,96
```

PERSONAL SOFTWARE

Phantoms Labirinto

Un avvincente gioco per il vostro VIC 20 in configurazione base

di *Ezio Bove*

Phantoms Labirinto è, come si può dedurre dal nome, un gioco di labirinto, che non mancherà di divertirvi.
Come prima cosa occorre osservare che nulla può essere aggiunto al gioco, considerato che a programma eseguito restano in memoria solo una settantina di byte liberi (il minimo indispensabile per permettere al VIC di eseguire il loop).
Esso può comunque subire delle piccole trasformazioni (nel caso che l'utente abbia il coraggio di farlo!) come ad esempio il colore dello schermo, il colore di fondo, quello degli oggetti disseminati nel labirinto e del labirinto stesso tramite le opportune Poke.
Con un po' di fantasia è possibile però modificare a proprio piacimento la forma del labirinto (a patto che questo sia delimitato).
Ricordo inoltre che la grafica del gioco è completamente in alta risoluzione, testi compresi, e consiglio dunque di digitare correttamente i Data dei caratteri e le Poke che via via incontrerete nel listato (questo per evitare di giocare con strane figure indecifrabili).

Un'ultima precisazione: il programma è completamente scritto in BASIC, ma la sua velocità di esecuzione (curata il più possibile) è perfettamente adeguata al tipo di game.

Il gioco

All'interno di un infernale labirinto avrete a che fare con uno scatenato fantasma che cercherà in tutti i

Listato 1. *Il programma Phantoms Labirinto.*

```
1 PRINT"[<1CLR>]":POKE36879,249:FORQ=7168
TO7679:POKEQ,0:NEXT
2 C1=7722:C=7908:FORK=0TO111:READAA:POKE7
568+K,AA:NEXT:Z1=30720
3 YY=1:Y=50:U=1:DS=36876:FORT=38400TO3890
5:POKET,6:NEXT:POKE36869,255
6 POKE36878,15:POKE51,247:POKE52,28:POKE5
3,247:POKE54,28:POKE55,247:POKE56,28
10 GOTO5000
12 FORJ=1TO750:NEXT:POKEDS,245:FORJ=1TO30
0:NEXT:POKEDS,0
15 TI$="000000":POKE7932+Z1,3:POKE7932,63
18 PRINT"[<20CRSR D>][<5CRSR R>][<1BLK>]9
:;[<1RVS>]"SC:PRINT"[<8CRSR L>]<=>[<1RV
S>]"YY
20 Z=INT(RND(0)*4)+1:IFZ=RTHEN20
30 IFZ=1ANDPEEK(C+A+22)=50THENR=Z:GOTO20
40 IFZ=2ANDPEEK(C+A-22)=50THENR=Z:GOTO20
50 IFZ=3ANDPEEK(C+A+1)=50THENR=Z:GOTO20
60 IFZ=4ANDPEEK(C+A-1)=50THENR=Z:GOTO20
70 POKEC+A,32
80 IFZ=1THENA=A+22:GOTO115
90 IFZ=2THENA=A-22:GOTO115
100 IFZ=3THENA=A+1:GOTO115
110 IFZ=4THENA=A-1
115 POKEC+A+Z1,4:POKEC+A,51
120 IFPEEK(C1+B+1)=50THEN130
125 IFPEEK(197)=37THENPOKEC1+B,32:B=B+1:G
OTO200
130 IFPEEK(C1+B-1)=50THEN140
135 IFPEEK(197)=29THENPOKEC1+B,32:B=B-1:G
OTO200
140 IFPEEK(C1+B-22)=50THEN150
145 IFPEEK(197)=17THENPOKEC1+B,32:B=B-22:
GOTO200
150 IFPEEK(C1+B+22)=50THEN200
155 IFPEEK(197)=33THENPOKEC1+B,32:B=B+22
200 IFPEEK(C1+B)<>32ANDPEEK(C1+B)<>52THEN
600
201 POKEC1+B+Z1,0:POKEC1+B,52:X$=RIGHT$(T
I$,2):X=VAL(TI$):IFX>YTHEN1000
210 S=S+1:IFS=10THENS=0:R=Z:GOTO20
220 W=W+1:IFW=13THENW=0:R=Z:GOTO20
230 R1=R1+1:IFR1>8-UTHENR1=0:GOTO300
240 PRINT"[<1HOME>][<21CRSR D>][<1BLK>]67
8[<1RVS>]"X$:GOTO30
300 IFZ=1ANDPEEK(C+A-22)<>50THENPOKEC+A-2
2+Z1,2:POKEC+A-22,53
310 IFZ=2ANDPEEK(C+A+22)<>50THENPOKEC+A+2
2+Z1,2:POKEC+A+22,53
320 IFZ=3ANDPEEK(C+A-1)<>50THENPOKEC+A-1+
Z1,0:POKEC+A-1,164
330 IFZ=4ANDPEEK(C+A+1)<>50THENPOKEC+A+1+
Z1,2:POKEC+A+1,53
350 POKEDS,230:FORJ=1TO30:NEXT:POKEDS,0:G
OTO30
600 IFPEEK(C1+B)<>53THEN630
610 FORJ=200TO130STEP-1:POKEDS+1,J:NEXTJ:
POKEDS+1,0:SC=SC-50:Y=Y-5:POKEC1+B,32
615 PRINT"[<1CRSR U>][<5CRSR R>]9:;[<1RV
S>]"SC:GOTO30
630 IFPEEK(C1+B)<>164THEN670
650 FORJ=240TO255:POKEDS,J:NEXT:POKEDS,0:
SC=SC+200:POKEC1+B,32
660 PRINT"[<1CRSR U>][<5CRSR R>]9:;[<1RV
S>]"SC:GOTO30
```

Seguito listato Phantoms Labirinto.

```
670 IFPEEK(C1+B)<>51THEN799
680 FORJ=220TO130STEP-1:POKEDS,J:NEXT:POK
EDS,0:U=U+1:IFU>5THENU=5
685 POKEC1+B,32:SC=SC+250
686 F=F+1:IFF>4THENF=1
687 IFF=1THENC1=7703
688 IFF=2THENC1=8118
689 IFF=3THENC1=8099
690 IFF=4THENC1=7722
691 PRINT"[<1CRSR U>][<5CRSR R>]9:;[<1RV
S>]"SC:PRINT"[<8CRSR L>]<=>[<1RVS>]"YY
695 A=0:B=0:YY=YY+1:TI$="000000":POKE7932
+Z1,3:POKE7932,63:GOTO20
799 IFD>2THEN30
800 FORJ=255TO245STEP-1:POKEDS,J:SC=SC+50
:FORH=0TO50:NEXTH:POKEDS,0
810 PRINT"[<1CRSR U>][<5CRSR R>]9:;[<1RV
S>]"SC:NEXTJ:D=D+1:TI$="000000":POKEC1+B,
32:GOTO30
1000 POKEC1+B+Z1,2:POKEC1+B,230:POKEDS+1,
220:FORJ=15TO0STEP-1:POKEDS+2,J:FORM=1TO3
00
1005 NEXTM,J:POKEDS+1,0:IFYY>=31THENPRINT
"[<11CRSR U>][<1CRSR R>][<1RVS>][<1BLK>]
   NEW  GAME":FORJ=1TO2500:NEXT:RUN
1010 PRINT"[<11CRSR U>][<1CRSR R>][<1RVS
>][<1BLK>]     GAME OVER"
1020 GETA$:IFA$="1"THENRUN
1030 IFA$="2"THENPRINT"[<1CLR>]":POKE3686
9,240:END
1040 IFA$<>"1"ORA$<>"2"THEN1020
3000 DATA255,255,255,231,231,255,255,255,
24,60,126,90,126,231,255,219
3010 DATA153,153,126,24,60,36,66,195,4,10
,16,56,124,116,124,56
3020 DATA2,0,250,34,34,34,34,0,0,0,219,25
0,171,138,139,0,0,0,152,24,0,24,152,0
3030 DATA0,0,238,136,232,40,238,0,0,0,247
,149,151,148,244,0,0,0,59,35,48,163,187,0
3040 DATA0,0,142,136,140,136,238,0,0,0,13
9,138,83,82,35,0,0,0,163,35,32,35,187,0
3050 DATA255,153,153,255,255,153,153,255
5000 FORT=7680TO7701:POKET,50:NEXT:FORT=8
120TO8141:POKET,50:NEXT
5010 FORT=7702TO8120STEP22:POKET,50:NEXT:
FORT=7723TO8141STEP22:POKET,50:NEXT
5020 FORT=7726TO7743:POKET,50:NEXT:FORT=7
771TO7785STEP2:POKET,50:NEXT
5030 FORT=7814TO7829:POKET,50:NEXT:FORT=7
858TO7874STEP2:POKET,50:NEXT
5040 FORT=7990TO8007STEP2:POKET,50:NEXT:F
ORT=8034TO8051:POKET,50:NEXT
5050 FORT=8078TO8095STEP2:POKET,50:NEXT:F
ORT=7743TO8047STEP22:POKET,50:NEXT
5060 POKE7941,32:POKE7734,32:POKE8039,32:
POKE8047,32
5070 FORT=7858TO7989STEP22:POKET,50:NEXT:
POKE7924,32:POKE7931,50:POKE7909,50
5080 POKE7953,50:POKE7954,50:POKE7910,50:
FORT=7912TO7917:POKET,50:NEXT
5090 FORT=7956TO7961:POKET,50:NEXT:POKE79
14,32:POKE7958,32
5100 FORT=7905TO7949STEP22:POKET,50:NEXT:
FORT=7907TO7951STEP22:POKET,50:NEXT
5110 POKE7809,32:POKE8029,32:GOTO12
```

modi di incastrarvi nei corridoi e di farvi esplodere.

Per far ciò si servirà di decine e decine di mine rosse con le quali vi bloccherà rapidamente ogni via d'uscita (a meno che voi non lo eliminiate del tutto).

Ma la sua perfidia non finisce qui, in quanto sarete costretti a seguirlo in ogni sua mossa e questo per due motivi: innanzitutto perché per poter passare al level successivo avrete a disposizione un solo minuto (che si decrementerà di 5 secondi per ogni mina presa).

In secondo luogo perché per far punti dovrete raccogliere i dollari da lui disseminati sempre più velocemente (level dopo level) sul piano di gioco.

Avrete comunque la possibilità, per sole tre volte in tutta la partita, di scampare il pericolo "tempo scaduto": vi è infatti, interno alla camera centrale del labirinto un bonus che vi riazzererà il tempo e vi regalerà molti punti.

A tal proposito vi ricordo che esistono otto e solo otto level di difficoltà e che dunque non dovrete preoccuparvi se l'indicatore di level ve ne segnalerà un numero più alto (si rimane sempre sull'ottavo livello).

Il modo per superare un level è prendere il fantasma, ma attenzione, perché proprio quando avrete creduto di afferrarlo egli potrebbe sfuggirvi dalle mani!

Voi sarete rappresentati da un omino che ad inizio gioco apparirà nell'angolo superiore a destra del labirinto.

Tale posizione cambierà comunque level dopo level secondo una sequenza che in tutta la partita resterà la stessa.

Potrete muovervi nelle quattro direzioni per mezzo di quattro tasti, e precisamente:

- tasto < per andare a sinistra;
- tasto > per andare a destra;
- tasto A per salire;
- tasto Z per scendere.

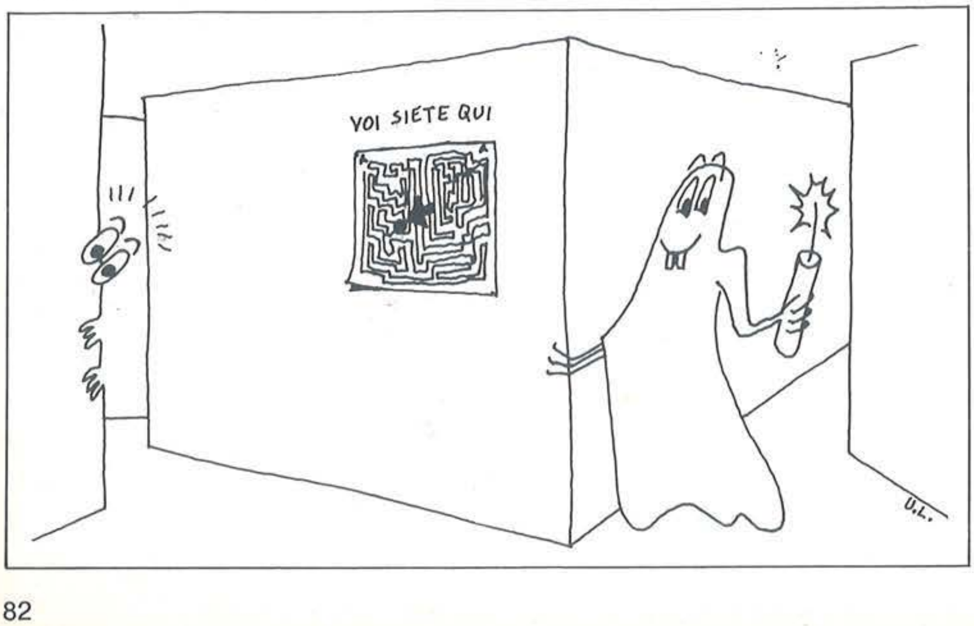
REMarks

1-3 - Inizializzazione variabili e co-

struzione caratteri speciali tramite istruzioni Read (linea 2) e Data (linee 3000-3050).
6 - Apertura volume e spostamento dei puntatori di fine memoria a 7415, dei puntatori area stringhe e lavoro stringhe per evitare il cancellamento dei caratteri speciali posti a partire dalla locazione 7568.
10 - Salta alla routine per la costruzione del labirinto.
12 - Ciclo di attesa e segnale acustico di inizio gioco.
15 - Azzeramento della variabile TI$.
18 - Visualizzazione delle scritte Score, Time, Level.
20-60 - In modo casuale viene fissata la direzione del fantasma.
Le linee 30-60 controllano se la posizione successiva è occupata dal bordo. In tal caso viene randomizzata una nuova direzione.
70 - Cancellazione della posizione del fantasma.
80-110 - In base alla generazione di valore casuale viene opportunamente incrementata o decrementata la variabile (B) della coordinata principale del fantasma.
115 - Colorazione e visualizzazione del fantasma.
120-155 - Spostamento dell'omino sul piano di gioco.
Viene letto il tasto premuto tramite il byte 197. Ed in particolare:

- il tasto 37 (segno di >) per andare a destra;
- il tasto 29 (segno di <) per andare a sinistra;
- il tasto 33 (Z) per scendere;
- il tasto 17 (A) per salire.

200 - Viene controllata una eventuale collisione tra l'omino e un qualsiasi oggetto posto nel labirinto (fantasma compreso).
In questo caso il programma salta alla routine 600.
201 - Visualizzazione dell'omino e confronto tempo scaduto.
In particolare vengono confrontate le variabili X (pari al valore di TI$) e Y opportunamente decrementata durante il gioco dalla collisione tra l'omino ed una mina.
210-220 - Per rendere più movimentato il gioco queste linee, in base all'incremento di due variabili, rimandano alla generazione di valori casuali per la direzione del fantasma.
230 - In base al valore della variabile U si salta meno (nei primi livelli di gioco) o più (man mano che questi aumentano) volte alla routine 300 per la deposizione degli oggetti (mine) da parte del fantasma.
240 - Visualizzazione del tempo e salto alla linea 30 (inizio loop principale).
300-350 - Routine per il posizionamento delle mine o dei dollari in base alla direzione del fantasma.
600-810 - Conseguenze delle collisioni tra l'omino e gli oggetti sparsi nel labirinto (fantasma compreso).
In particolare:
600-615 - Collisione tra omino e mina:
viene decrementata la variabile SC per il punteggio e la variabile Y per il tempo a disposizione.
630-660 - Collisione tra l'omino e il simbolo del dollaro: viene incrementato il solo punteggio.
670-695 - Collisione tra l'omino e il fantasma:
viene incrementata la variabile U per velocizzare il deposito di mine e di dollari da parte del fantasma.
Incremento del punteggio.
Incremento della variabile F per il posizionamento iniziale dell'omino in uno dei quattro angoli del labirinto.
Azzeramento delle variabili A e B (variabili principali delle coordinate dell'omino e del fantasma).
Incremento della variabile YY per la visualizzazione del livello di gioco.
Azzeramento del tempo (TI$) e rivisualizzazione del bonus al centro del campo di gioco.
799-810 - Collisione tra l'omino e il bonus: viene controllato alla linea 799 se il bonus è stato già preso tre volte: in tal caso il programma ritorna al loop principale.
Altrimenti: viene incrementato il punteggio e azzerata la variabile TI$.
1000-1040 - Routine Fine Gioco: se si è superato il 30° livello di gioco (in base alla variabile YY) viene visualizzata la scritta "New Game" e si riparte con una nuova partita.
In caso contrario apparirà la scritta "Game Over" ed il gioco sarà concluso.
Le linee 1020-1040 consentono, tramite istruzione Get, l'uscita definitiva dal programma o l'inizio di una nuova partita.
Il Poke 36869,240 riabilita la mappa caratteri standard.
3000-3050 - Dati caratteri speciali.
5000-5110 - Costruzione labirinto.■



Facile

CALCOLARE...

È facile con MULTIPLAN.
Questo programma per la gestione del "foglio elettronico" trasforma il vostro personal computer in un prestigioso calcolatore che utilizzerete senza problemi.
Istruzioni, comandi e ampia documentazione.
Incolonnamento variabile.
Indirizzamento relativo o assoluto. Tutto facilita il lavoro. Completo di guida molto chiara che potete far apparire, a richiesta, sullo schermo.

MODIFICARE...

Desiderate cambiare dei parametri? Multiplan ricalcola automaticamente tutto ciò che ne deriva. La medesima cosa su più fogli di calcolo che potrete legare tra loro a volontà. Integrando funzioni logiche, funzioni statistiche Multiplan si rivela il "foglio elettronico" più potente sul mercato.

DECIDERE...

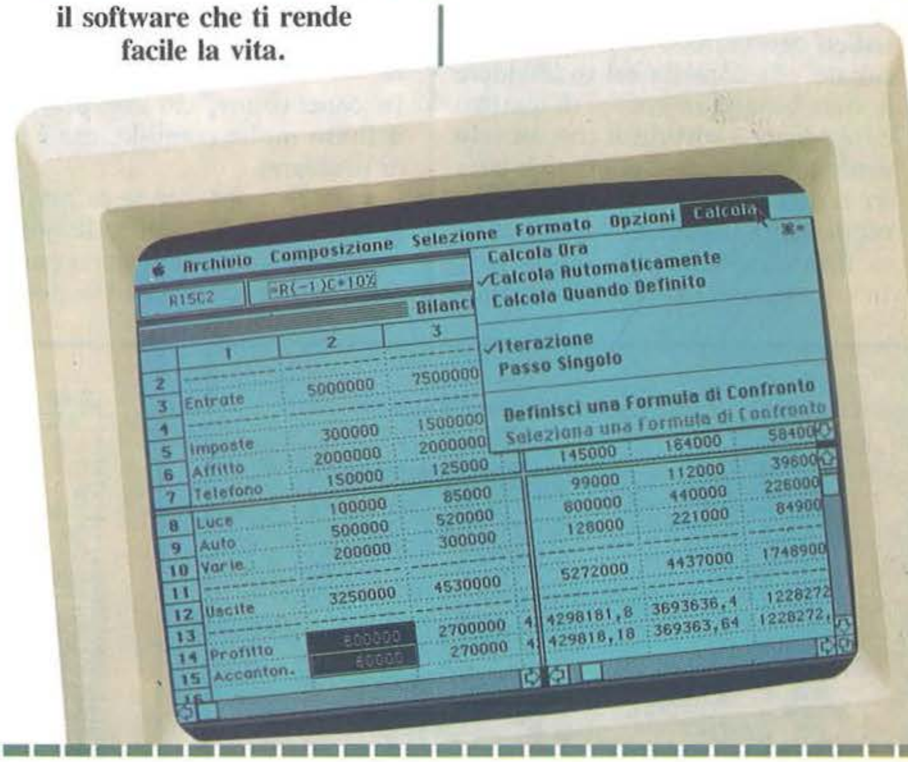
Con Multiplan avrete sotto gli occhi tutte le cifre per prendere decisioni oggettive. Eletto programma dell'anno, già uno dei best-sellers mondiali del 1983, Multiplan, è ora disponibile in Italiano.

con Multiplan

PER APPLE MACINTOSH
IN EDIZIONE ITALIANA

MICROSOFT®

il software che ti rende facile la vita.



DISTRIBUITO IN ITALIA
ESCLUSIVAMENTE DA

J.soft EDITRICE

20124 Milano - Via Rosellini, 12
Tel. (02) 6888228-683797-6880841/2/3

Potete acquistare Multiplan - edizione italiana - presso i migliori rivenditori Apple o riceverlo direttamente da J.*soft*, compilando ed inviando il coupon sotto riportato.

Inviare a J.*soft*, - Via Rosellini, 12 - 20124 Milano

Ordino n° .... Multiplan in italiano per Apple Macintosh - cod. DMOPM02 al prezzo di L. 638.000 cad. (IVA e spese di spedizione incluse).

Scelgo la seguente forma di pagamento:
□ pagherò in contrassegno al postino
□ assegno allegato di L. ....

Nome .... Cognome ....

Via ....

C.A.P. .... Città .... Prov. ....

Desidero fattura – n° Partita IVA ....

Data .... Firma ....

Super Converter per Spectrum

Due micro-utility ... veramente utili!!

di *Stefano Cerutti*

Quando per la prima volta, nella "preistoria dell'informatica" prima dell'adozione di linguaggi ad alto livello, si cominciarono ad adottare sistemi di numerazione atti a facilitare il lavoro del programmatore, un grande aiuto fu dato dal sistema in base sedici, denominato appunto esadecimale, che consiste nel suddividere le cifre binarie in gruppi di quattro cifre (digit) e sostituirli con un solo simbolo: per questo si usano le cifre da zero a nove, seguite dalla A che rappresenta il dieci, la B come undici, fino alla F che vale quindici.
In seguito, con l'avvento di linguaggi come il BASIC, il Pascal e il FORTH, il programmatore fu dispensato dall'ingrato compito di manipolare continuamente interminabili serie di uno e zero, soggette oltretutto a facili errori di svista da parte dell'uomo, non avvezzo a lavori di altissima precisione e ripetitività.
Ora il problema viene lasciato ai progettisti di sistemi operativi e di interpreti.
Ai giorni nostri chi si cimenta con l'Assembly, è costretto a continue consultazioni delle tavole di conversione, per ricavare l'equivalente decimale degli indirizzi esadecimali forniti dal programma assemblatore.
In conclusione, ciò che prima era definito molto comodo, ora è fonte di problemi.

I neofiti, che per la prima volta approdano sul "pianeta linguaggio macchina", apprezzeranno particolarmente le utility che sto per porre.
Il caricatore BASIC del listato uno genera due programmini in linguaggio macchina che permettono la conversione istantanea sia da decimale a esadecimale che viceversa.
Per le istruzioni di caricamento, occorre riferirsi alla tabella 1.

Alcuni consigli

Poichè l'algoritmo è ridotto all'essenziale, non sono state inserite le istruzioni atte a controllare l'overflow della routine Dec To Hex: se vengono inseriti valori superiori a 65.535, verranno visualizzati risultati errati.
Mentre la routine Hex To Dec deve ricevere sempre quattro cifre esadecimali, l'altra routine percepisce anche la pressione del tasto Enter, per troncare i numeri inferiori a cinque cifre decimali.
Occorre prestare attenzione anche

Listato 1. *Il programma Super Converter.*

```
   5 REM ________________________
  10 CLEAR PEEK 23676*256+71
  20 POKE 23675,72: LET a=USR "a
": LET x=PEEK 23676: LET c=0
  30 FOR f=a TO a+183: READ n
  40 POKE f,n: LET c=c+n: NEXT f
  50 IF c<>20474+11*x THEN PRINT
 "Hai battuto male qualcosa:
  Controlla bene i DATA !!!": ST
OP
  60 PRINT " PRINT USR USR ""a""
 = Hex TO Dec"
  70 PRINT '" PRINT USR USR ""j"
" = Dec TO Hex"
  80 PRINT '''TAB 8;"Super Conve
rter"
  90 PRINT 'TAB 13;"READY"
  91 REM
  92 REM
  93 REM   DATA per L.M.
  94 REM
  95 REM
 100 DATA 33,253,x,6,4,205,234
 110 DATA x,205,132,x,205,27,45
 120 DATA 48,10,254,103,48,244
 130 DATA 254,97,56,240,214,32
 140 DATA 245,215,241,214,48,254
 150 DATA 10,56,2,214,7,237,111
 160 DATA 120,203,31,48,1,43,205
 170 DATA 224,x,16,214,62,2,205
 180 DATA 1,22,237,75,252,x,201
 190 DATA 175,50,8,92,58,8,92
 200 DATA 167,40,250,201,0,239
 210 DATA 160,56,6,5,205,234,x
 220 DATA 205,132,x,79,197,205
 230 DATA 34,45,193,48,8,254,13
 240 DATA 32,241,6,1,24,10,197
 250 DATA 239,1,164,4,15,56,193
 260 DATA 121,215,205,224,x,16
 270 DATA 222,205,162,45,237,67
 280 DATA 252,x,205,122,x,6,4
 290 DATA 33,253,x,175,237,111
 300 DATA 198,48,254,58,56,2,198
 310 DATA 7,215,120,203,31,48,1
 320 DATA 43,16,236,205,248,31
 330 DATA 217,17,100,0,213,225
 340 DATA 205,181,3,217,217,175
 350 DATA 205,1,22,217,205,7,25
 360 DATA 62,8,215,217,33,88,39
 370 DATA 217,201,0,0,0,0
 400 REM ________________________
```

al cursore, che deve essere in modo "L" prima di effettuare la chiamata con Print Usr Usr "a", altrimenti non verranno accettate le cifre esadecimali non numeriche (da A a F). Questa routine di conversione da esadecimale a decimale ha il vantaggio di sostituire il risultato al valore della Usr: se è necessario salvare su nastro i byte da A0CA fino a A200, per avere i parametri da passare all'istruzione Save, si può dare una Print Usr Usr "a"—Usr Usr "a"+1 per ottenere la lunghezza, e Print Usr Usr "a" per lo start, tenendo presente che l'input avviene da sinistra verso destra, cioè nel nostro caso dobbiamo inserire prima A200 e poi di seguito A0CA.
Un'ultima avvertenza, ma è la più importante: se vi capita di dover convertire una lunga serie di numeri e vi viene la malsana idea di eseguire una linea 1 Print Usr Usr "a": Go To 1 oppure "j", non fatelo assolutamente, perché non riuscireste più a ritornare al BASIC e sarete costretti a spegnere tutto perdendo i dati-RAM.

Le chiamate vanno sempre fatte con un comando diretto Print Usr Usr "a" o "j".
Se doveste incappare malauguratamente in tale "endless loop", potreste tentare di uscirne continuando a premere tasti, e, dato che i residui sulla lower part non vengono cancellati, aspettare una richiesta di "scroll?" e rispondere con Break.

Uno sguardo al disassemblato

Per diminuire l'occupazione di memoria sono stati utilizzati molti sottoprogrammi della ROM e istruzioni di Restart (Rst), impiegando persino il calcolatore floating point, il quale possiede un "linguaggio interno", memorizzato sotto forma di

Figura 1. *Disassemblato dei due algoritmi: tutti i valori sono in esadecimale.*

ORG FF48 (ORG 7F48 per il 16 Kbyte)

```
FF48   HEX TO DEC:LD HL, MEM2
FF4B              LD B,04
FF4D              CALL L0
FF50           L3:CALL L1
FF53              CALL 2D1B
FF56              JR NC,L2
FF58              CP 67
FF5A              JR NC,L3
FF5C              CP 61
FF5E              JR C,L3
FF60              SUB 20
FF62           L2:PUSH AF
FF63              RST 10
FF64              POP AF
FF65              SUB 30
FF67              CP 0A
FF69              JR,CL4
FF6B              SUB 07
FF6D           L4:RLD
FF6F              LD A,B
FF70              RR A
FF72              JR NC,L5
FF74              DEC HL
FF75           L5:CALL 16
FF78              DJNZ L3
FF7A           LB:LD A,02
FF7C              CALL 1601
FF7F              LD BC,(MEM1)
FF83              RET
FF84           L1:XOR A
FF85              LD (5C08),A
FF88           L7:LD A,(5C08)
FF8B              AND A
FF8C              JR Z,L7
FF8E              RET
FF8F              NOP
FF90   DEC TO HEX:RST 28
FF91              DB A0
FF92              DB 38
FF93              LD B,05
FF95              CALL L0
FF98           L9:CALL L1
FF9B              LD C,A
FF9C              PUSH BC
FF9D              CALL 2D22
FFA0              POP BC
FFA1              JR NC,L8
FFA3              CP 0D
FFA5              JR NZ,L9
FFA7              LD B,01
FFA9              JR LA
FFAB           L8:PUSH BC
FFAC              RST 28
FFAD              DB 01
FFAE              DB A4
FFAF              DB 04
FFB0              DB 0F
FFB1              DB 38
FFB2              POP BC
FFB3              LD A,C
FFB4              RST 10
FFB5           LA:CALL L6
FFB8              DJNZ L9
FFBA              CALL 2DA2
FFBD              LD (MEM1),BC
FFC1              CALL LB
FFC4              LD B,04
FFC6              LD HL,MEM2
FFC9           LE:XOR A
FFCA              RLD
FFCC              ADD A,30
FFCE              CP 3A
FFD0              JR C,LC
FFD2              ADD A,07
FFD4           LC:RST 10
FFD5              LD A,B
FFD6              RR A
FFD8              JR NC,LD
FFDA              DEC HL
FFDB           LD:DJNZ LE
FFDD              CALL 1FF8
FFE0           L6:EXX
FFE1              LD DE,0064
FFE4              PUSH DE
FFE5              POP HL
FFE6              CALL 03B5
FFE9              EXX
FFEA           L0:EXX
FFEB              XOR A
FFEC              CALL 1601
FFEF              EXX
FFF0              CALL 1907
FFF3              LD A,08
FFF5              RST 10
FFF6              EXX
FFF7              LD HL,2758
FFFA              EXX
FFFB              RET
FFFC         MEM1:NOP
FFFD         MEM2:NOP
FFFE              NOP
FFFF              NOP
```

Super Converter per Spectrum

• Dai un Print Usr 0, perché la variabile di sistema Udg non deve essere alterata in quanto il programma stabilisce in base al suo valore se sta girando su uno Spectrum 16 Kbyte o su un 48 Kbyte.
• Digita il listato 1.
• Dai un Run fino a quando non riceverai più il messaggio di errore.
• Dai un Clear: Save "Converter" Line 1 e registra una copia su nastro per i successivi utilizzi.
• Ora puoi anche dare un New, dal momento che il caricatore Basic è inutile e l'assemblato di figura 1 si trova sopra a RAMtop e quindi non viene cancellato.
• Ora puoi caricare in memoria il programma che devi mettere a punto, ma stai attento che la variabile Udg (23675 e 23676) non venga modificata in nessun modo, altrimenti cambia il valore di Usr "a" e Usr "j".
Se devi proprio manipolarla, tieni presente che puoi effettuare le chiamate alla sub Hex to Dec anche con Print Usr 32584 per il 16 Kbyte o Usr 65352 per il 48 Kbyte e alla Dec to Hex con Usr 32656 o 65424.
• Ricorda che non hai a disposizione i caratteri User-defined.
• Ricorda che sono stati generati due Udg aggiuntivi, pertanto il valore di Usr "a" e seguenti è più basso del solito (-16).

DB (op-code) dopo l'istruzione Rst 40.
Per chi fosse interessato a modifiche eventuali o semplicemente volesse approfondire la conoscenza della dinamica del programma di figura 1, ecco alcune informazioni.
La routine con label "L1" non ritorna fino a che non viene premuto un tasto, e riporta il suo codice ASCII nell'accumulatore.
La "L6" emette un beep, mentre la routine su ROM che inizia a 2D22 deposita la cifra il cui codice si trova nell'accumulatore, sul calculator stack e ritorna con il carry flag settato se non sta trattando con un codice di una cifra (da 48 a 57).
La 2DA2 passa l'ultimo valore del calculator stack ai registri BC: carry set per numeri maggiori di 65535 e zero set per numeri positivi.
La 1FF8 equivale a Print Chr$ 13, ed infine la 1907 stampa una C lampeggiante.
I due programmini sono "compressi" negli Udg, ma poiché occupano più dei 168 byte disponibili, è stato necessario creare due Udg aggiuntivi (16 byte) abbassando tutto il blocco.
Per questo ed altri motivi, è consigliabile caricare sempre prima Converter, dare un New e poi memorizzare l'eventuale Assembler con codice sorgente.

Considerazioni finali

Durante il loro utilizzo, le routine presentate in queste pagine non limitano nè sottraggono memoria ai programmi BASIC o linguaggio macchina, e sono immuni da New.
Con queste utility spero di essere riuscito ad attenuare le difficoltà che ostacolano il lavoro dei programmatori linguaggio macchina; comunque, per i programmatori BASIC sono in arrivo grosse sorprese.

COMPUTER SHOW 85

19 - 23 APRILE 1985 - FIERA DI MILANO

Ogni anno circa due milioni di persone visitano la Fiera Campionaria di Milano. Dal 1985, ad attenderle, ci sarà COMPUTER SHOW il nuovo Salone interamente dedicato all'informatica per i giovani, la scuola, la famiglia moderna. Sicuramente sarà il più grande appuntamento dell'anno.

Perché non esserci?



la mostra per:

l'hobby
la didattica
i videogiochi

Segreteria:
E.P.I. - ENTE PROMOZIONE INFORMATICA
Via Marochetti, 27 - 20139 Milano
Telefoni (02) 56.93.973 - 53.98.267

Apple: pagine video e dintorni

Utilizziamo Peek e Poke per i nostri programmi BASIC

di *Gianfranco Pisani*

Sono ormai parecchi anni che abbiamo sviluppato tecniche alternative per la gestione video dell'Apple e l'apparizione sul numero 14 di Aprile dell'articolo di Claudio Poma (*Peek, Poke e lo schermo dell'Apple* a pag. 24) ci ha dato l'idea di sviluppare queste note per approfondire ulteriormente l'analisi della pagina video e dei suoi segreti.
L'articolo in questione ripresentava in versione Apple il Gioco d'inseguimento già pubblicato in precedenza; ora ci proponiamo di riscriverlo abolendo tutte le Vtab, Htab, e Print per lasciar posto solo ai Poke (per scrivere) ed ai Peek (per leggere), laddove il vecchio programma usava solo la Peek nelle linee 184-187.
Introdurremo inoltre un diverso algoritmo per la ricerca della locazione di memoria all'interno della pagina video in sostituzione di quello usato dal programma, che risulta un po' lento per la presenza di ben 3 test (500-520) che possono agevolmente essere eliminati.

Prima di lasciare il programma così riscritto in balia della fantasia di ciascuno per modifiche e personalizzazioni varie occorrerà poi eliminarne un difetto di base: il Computer lanciato all'inseguimento ha la facoltà, in determinate situazioni, di rompere il muro di cinta esterno aprendo un pericoloso varco attraverso il quale un'abile lepre può uscire dal recinto ed impegnarsi in una fuga senza frontiere anche al di là dei confini della stessa pagina video, dove sono in agguato i micidiali (per il programma) Illegal Quantity Error al soldo di istruzioni tipo Htab 41 e Vtab 0 ...!
Ma andiamo con ordine: prima un po' di teoria sul video, poi il programma riscritto con Peek e Poke quindi l'analisi e l'eliminazione dei suoi errori con qualche piccola innovazione personale e in ultimo, come dicevamo prima, a ciascuno il suo inseguimento personale.

Un po' di teoria

Il video presenta in Text Mode 960 caratteri il cui codice ASCII deve essere conservato in altrettante locazioni di memoria. Essendo questa organizzata, per ragioni di indirizzamento, in pagine di 256 byte (esadecimale FF) occorrono ben 4 pagine (4, 5, 6 e 7 per un totale di 1.024 byte) per gestire l'intero video. Rimangono tuttavia libere 64 locazioni che l'Apple riserva al DOS e alle ROM presenti sulle schede di interfacciamento.
La seguente tabella mostra l'organizzazione dell'area di memoria compresa tra le locazioni 1024 ($400) e 2047 ($7FF):

Riga 1 1024-1063	Riga 9 1064-1103
Riga 2 1152-1191	Riga 10 1192-1231
Riga 3 1280-1319	Riga 11 1320-1359
Riga 4 1408-1447	Riga 12 1448-1487
Riga 5 1536-1575	Riga 13 1576-1615
Riga 6 1664-1703	Riga 14 1704-1743
Riga 7 1792-1831	Riga 15 1832-1871
Riga 8 1920-1959	Riga 16 1960-1999
Riga 17 1104-1143	Speciali 1144-1151
Riga 18 1232-1271	Speciali 1272-1279
Riga 19 1360-1399	Speciali 1400-1407
Riga 20 1488-1527	Speciali 1528-1535
Riga 21 1616-1655	Speciali 1656-1663
Riga 22 1744-1783	Speciali 1784-1791
Riga 23 1872-1911	Speciali 1912-1919
Riga 24 2000-2039	Speciali 2040-2047

Una lettura per righe della tabella permette di scorrere la memoria sequenzialmente mentre una lettura per colonne fa altrettanto del video.
Dall'analisi delle colonne otterremo l'algoritmo di conversione tra coordinate video (V da Vtab e H da Htab) e locazione di memoria (M) e dalle righe quello inverso.
La tabella presenta 8 righe comprendenti ciascuna tre righe del video (40 byte) e un gruppo di 8 byte speciali: in seguito ci riferiremo a tale riga di tabella col termine "gruppo di 128 byte" per non ingenerare confusione tra le righe di tabella e quelle del video. Definiamo con G l'indice di Gruppo (da 0 a 7).
Come già visto nell'articolo di Poma le tre righe video di ciascun gruppo appartengono a tre diverse fasce (colonne di tabella) in cui si divide il video: la prima dal rigo 1 al 8, la seconda da 9 a 16, la terza da 17 a 24.
Il *primo* gruppo di 128 byte riunisce quindi le *prime* righe di ciascuna fascia (1-9-17) e così di seguito il *secondo* gruppo conterrà le *seconde* righe (2-10-18) ecc. Definiamo con F l'indice di Fascia (da 0 a 2) e con B(F) l'indirizzo base di ciascuna fascia. Il valore di F sarà:

Apple: pagine video e dintorni

$$F = INT((V - 1) / 8)$$

Ora la locazione M cercata si ottiene aggiungendo H - 1 all'indirizzo BASE B(V) di ciascuna riga. Tenendo quindi conto della tabella 1 (valori di B (F)), e le seguenti formule:

$$M = B(V) + H - 1$$
$$B(V) = B(F) + G \star 128$$
$$B(F) = 1024 + (F \star 40)$$
$$G = V - 1 - (F \star 8)$$

Tab 1	Prima fascia	Seconda fascia	Terza fascia
B(F)	1024	1024+40	1024+80
G	V—1	V—1—8	V—1—16

da cui si ha:

$$M = 1024 + (F \star 40) + (V - 1 - (F \star 8)) \star 128 + H - 1$$
$$M = 1024 + (F \star 40) + (V \star 128) - 128 - (F \star 1024) + H - 1$$
$$M = 895 + (V \star 128) - (F \star 984) + H$$

sostituendo quindi il valore di F otteniamo il seguente algoritmo finale:

$$M = 895 + (V \star 128) - 984 \star INT((V - 1) / 8) + H$$

Il programma numero 1, chiamato Vh-Mem, dimostra come si possa riempire il video sequenzialmente per righe (For di linea 20 e 30) sfruttando l'algoritmo (linea 40) insieme alla istruzione di Poke (linea 50). Si nota in particolare che le Poke che scrivono sulle righe 23 e 24 non provocano alcuno scroll del video.

Listato 1. *Riempie sequenzialmente il video con linee.*

```
1  REM *
2  REM * G. PISANI
3  REM * PROG. 1 - VH-MEM
4  REM *
10  HOME
20  FOR V = 1 TO 24
30  FOR H = 1 TO 40
40 M = 895 + (128 * V) - 984 *  INT
     ((V - 1) / 8) + H
50  POKE M,164
60  NEXT H,V
70  END
```

L'algoritmo di Vh-Mem è stato ricavato partendo dal valore di F (fascia) in funzione di V noto e dalla legge di variazione di B(V) all'interno di ciascuna colonna di tabella (fascia). Ora partendo dal valore di G (gruppo) in funzione di M noto e dalla legge di variazione di R (posizione relativa di M nel suo gruppo) all'interno di ciascuna riga di tabella (gruppo) troveremo l'algoritmo inverso. I valori di G e di B(G) (indirizzo base di un gruppo di 128 byte o riga di tabella) e di R, così come definito, sono quindi:

$$G = INT((M - 1024) / 128)$$
$$B(G) = 1024 + G \star 128$$
$$R = M - B(G) = M - 1024 - G \star 128$$

ora R può variare da 0 a 119 e ci permette di determinare la colonna di tabella a cui appartiene M: infatti Int (R / 40) vale rispettivamente 0, 1, 2 per le tre fasce (si rivede quindi F). Già sappiamo che:

$$G = V - 1 - (F \star 8)$$

da cui ricaviamo V:

$$V = G + (F \star 8) + 1 = G + INT(R / 40) \star 8 + 1$$

il valore di R si presta facilmente a darci anche H:

$$H = R + 1 - (F \star 40) = R + 1 - INT(R / 40) \star 40$$

L'algoritmo così trovato è stato riassunto (linee 110-140) nel programma 2, chiamato Mem-Vh, che scrive il video (linea 150 con If verificato) nell'ordine dettato dalla sequenza della memoria (For di linea 100) scartando le locazioni speciali (If di linea 150 non verificato) di cui alla fine dà le coordinate ipotetiche (linee 200-270) che, come si vedrà, sono tutte illegali.

Possibili usi di un simile algoritmo: ricercare sull'uscita video di un programma una stringa di caratteri particolari e segnalarne poi le coordinate precise all'utente, usare il video come una matrice di input/output senza necessità di memorizzare in vettori ma con la comodità di vedere lo stato della matrice nella sua evoluzione dinamica (V e H diventano quindi gli indici della matrice).

Rimane da vedere la funzione delle famose 64 locazioni speciali: attenzione non toccatele mai se non siete proprio sicuri di quel che fate!

Esse contengono dati come la slot in uso, la configura-

zione del sistema, il tipo di stampante, gli indirizzi base di alcune aree di memoria destinate alla gestione degli I/O ecc., e se vi venisse in testa di gestire tutto il video come un unico blocco di memoria da trasferire e sostituire con altro (gestioni veloci delle maschere video, animazioni varie, salvataggi su disco ecc.) ricordatevi prima di salvare almeno i più importanti di questi registri e rimetterli dopo al loro posto altrimenti potreste trovarvi in guai seri. Per addentrarci in questo campo si dovrebbe scrivere un articolo a parte, sempre se sarete riusciti a digerire questo, tuttavia, mosso a compassione, vi farò un esempio pratico più avanti.

Avendo dissertato anche sulle locazioni speciali che si possono considerare come i dintorni citati nel titolo, l'argomento della pagina video sembrerebbe esaurito qui, invece è proprio qui, ai confini del visibile, che si apre una porta verso un nuovo mondo sconosciuto (o quasi) e sommerso (normalmente) sotto cumuli di programmi: di lui il manuale sembra dire "c'è ma non si tocca", questo mondo è la seconda pagina video e la sua porta è Poke - 16299,0.

Provate a dare questo comando dalla tastiera e subito il video cambia aspetto visualizzando l'area di memoria che va da 2048 ($800) a 3071 ($11FF).

I 960 caratteri visualizzati sono da considerarsi Random; il sistema, infatti, alloca i programmi a partire dalla locazione 2049 ($801), riempiendo le locazioni successive prima con i codici generati dal BASIC e poi con le variabili e le matrici: ora ciascuno dei 256 valori che può assumere una locazione genera un ben definito carattere video non avendone, però, il significato logico. Digitando un New seguito da Return noterete i primi tre caratteri del video che cambiano in "@" ma in realtà abbiamo solo azzerato le locazioni 2048, 2049 e 2050 ($800, $801 e $802) creando un End of Program (un successivo List, infatti, non sortirà alcun listato). Ah, dimenticavamo, mentre digitate il New non lo vedrete a video perché l'Apple lo sta scrivendo sulla pagina 1 e noi, per ora, non lo possiamo vedere. È evidente, quindi, che il solo Poke -16299,0 non ci permetterà di usare la pagina 2, a meno di non voler scrivere sul programma distruggendolo.

Per prima cosa ritorniamo in pagina 1 (digitando Poke -16300,0), sempre al "buio": dopo il Return vedrete ricomparire l'istruzione insieme al famoso New scomparso in precedenza. Vi invitiamo a questo punto a fare delle prove: potrete notare che la presenza sul video della pagina 2 non impedisce al sistema di funzionare egualmente, infatti digitando al "buio" un comando errato si avvertirà il beep usuale mentre l'avviso di errore sarà regolarmente emesso in pagina 1, un comando di PR # attiverà la stampante e un List stamperà il programma, un Run lo eseguirà con emissione video sempre in pagina 1.

Possiamo anche modificare il programma caricato digi-

Listato 2. *Il programma Mem-Vh, che scrive il video.*

```
1   REM *
2   REM * G. PISANI
3   REM * PROG. 2 - MEM-VH
4   REM *
40   DIM MS(64),VS(64),HS(64)
50   HOME
100   FOR M = 1024 TO 2047
110 G =  INT ((M - 1024) / 128)
120 R = M - 1024 - 128 * G
130 V = G + 1 +  INT (R / 40) * 8

140 H = R + 1 -  INT (R / 40) * 4
    0
150   IF V < 25 THEN  VTAB V: HTAB
    H: PRINT "$";: GOTO 170
160 S = S + 1:MS(S) = M:VS(S) = V
    :HS(S) = H
170   NEXT M
190   FOR A = 1 TO 1000: NEXT
200   HOME
210   PRINT "NUM    MEM    VTAB    HT
    AB": VTAB 3: POKE 34,2
220   FOR A = 0 TO 7
230   FOR S = 1 + A * 8 TO 8 + A *
    8
240   PRINT S;: HTAB 6: PRINT MS(S
    );"      ";VS(S);"      ";HS(S
    )
250   NEXT S
260   PRINT : INPUT "PREMI RETURN
    ";A$: PRINT
270   NEXT A
280   POKE 34,0: HOME : END
```

Listato 3. *L'area di memoria della pagina 2 è liberata.*

```
1   REM *
2   REM * G. PISANI
3   REM * PROG. 3 - VIDEO2
4   REM *
100   POKE 3072,0: POKE 3073,0: POKE
    3074,0
110   POKE 104,12: POKE 103,1
120   POKE 106,12: POKE 105,4
130   POKE 108,12: POKE 107,4
140   POKE 110,12: POKE 109,4
150   POKE 176,12: POKE 175,4
160   END
```

tando una istruzione nuova o un Del (se la modifica è all'inizio si potrà notare un cambiamento sul video). Tornando in pagina 1 ritroveremo traccia di tutte le ultime operazioni eseguite al buio.

La spiegazione di ciò sta nel fatto che l'Apple, come tutti i computer, non legge o scrive un video, ma legge una tastiera e scrive in memoria (prova evidente di ciò è l'istruzione Get che non ha effetti diretti sul video), è quindi possibile, in via di principio, lavorare anche a video spento.

La pagina 2 è strutturata in gruppi e fasce esattamente come la prima. Unica differenza è che le 64 locazioni in soprannumero non sembrano avere funzioni specifiche. La relativa tabella si ottiene da quella già vista per la pagina 1 aggiungendo 1024 a tutti gli indirizzi.

Il programma 3, chiamato Video2, libera l'area di memoria della pagina 2 dai compiti affidatigli dal sistema, spostando all'inizio della pagina 12 di memoria l'inizio dei programmi caricati da tastiera o da dischetti (confronta articoli di Staderini *Rilocare l'Apple BASIC e altre cosucce Bit* 38 e di Azzali *Apple a mezzadria ovvero: due programmi in memoria Bit* 44). Potremo ora vedere come si può scrivere e leggere la pagina 2 senza i problemi esposti; riprovate a rifare, ad esempio, il New, non succederà nulla alle prime tre locazioni perché ora il New agisce a partire da 3072 ($1200).

Introduciamo, dopo V ed H, una terza coordinata: P

Listato 4. *Riempie le due pagine video sequenzialmente.*

```
1   REM *
2   REM * G. PISANI
3   REM * PROG. 4 - VHP-MEM
4   REM *
100  HOME
110  FOR V = 1 TO 24
120  FOR P = 1 TO 2
130  FOR H = 1 TO 40
140 M = (P * 1024) - 129 + (128 *
     V) - 984 * INT ((V - 1) / 8
     ) + H
150  POKE M,176 + P
160  NEXT H
170  POKE ( - 16298 - P),0
180  NEXT P
190  NEXT V
200  FOR A = 1 TO 3000: NEXT
210  POKE - 16299,0
220  FOR A = 1 TO 3000: NEXT
230  POKE - 16300,0
240  HOME : END
```

che sta per Pagina e vale 1 o 2 per le due pagine. L'algoritmo di Vh-Mem diventa generale se al valore di M trovato si somma (P - 1) ★ 1024, cioè:

M = 895 + (V ★ 128) - 984 ★ INT((V - 1)/8) + H + (P - 1) ★ 1024
M = (P ★ 1024) - 129 + (V ★ 128) - 984 ★ INT((V - 1)/8) + H

Ora dopo aver fatto girare Video2 caricate il programma 4, chiamato Vhp-Mem, (inizierà in 3073): questo programma riempie le due pagine video sequenzialmente per righe (For di linea 110-130) scrivendo con la Poke (150) i caratteri 1 e 2 nelle locazioni via via calcolate con l'algoritmo (140).

L'istruzione 170 si incarica di cambiare pagina alla fine di ogni riga.

Proviamo ora a generalizzare l'algoritmo inverso, quello cioè di Mem-Vh: la soluzione aprirà la strada alla possibilità di scrivere la pagina due con un normale Print unito all'uso di Vtab e Htab e di operare anche con Input e Get, anche se, come vedremo ci saranno delle differenze operative a volte vantaggiose, a volte no.

Diciamo che il set di istruzioni Applesoft necessiterebbe di un nuovo comando che definiremo con Ptab P, questo dovrebbe lavorare come Vtab e Htab, cioè dovrebbe dare la terza coordinata video del cursore indirizzandola alla pagina 1 o 2. Ora P dipende da M noto:

P = 1 + (M (>) 2047)

trovato P l'algoritmo già definito per V ed H potrà funzionare ancora con una semplice modifica: il valore di partenza per il calcolo di G ed R era (M - 1024) che dava la posizione relativa di M rispetto all'indirizzo di inizio pagina, ora tale posizione diventa (M - P ★ 1024). Il programma 5, chiamato Mem-Vhp, ci mostra l'algoritmo (linee 105 e 110-140) così generalizzato: il For di linea 100 ora va da 1024 a 3071 e la linea 108 si incarica di girare pagina appena P = 2; la linea 150 al solito stampa il carattere nella posizione V, H, P (1 o 2 a seconda della pagina) ma al posto di Ptab P (che non esiste) compare un Gosub 300.

Tutti sanno che le locazioni 32, 33, 34 e 35 gestiscono le finestre sul video, molti che le locazioni 36 e 37 contengono i valori correnti di Htab e Vtab diminuiti di una unità, pochi, ancora, che le locazioni 38 e 39 contengono l'indirizzo della corrente subroutine di input (generalmente Keyin). Ora le locazioni 40 e 41 hanno una funzione che non sempre compare sui manuali e che è molto interessante: rappresentano l'indirizzo iniziale di una riga video, quella a cui appartiene la posizione corrente del cursore. Ad ogni Vtab, Print senza il ";" finale, o comunque a cavallo di due righe, Input e tutte le istruzioni che variano la riga del cursore il sistema aggiorna

Apple: pagine video e dintorni

Listato 5. *IL programma Mem-Vhp, con algoritmo inverso al precedente.*

```
1  REM *
2  REM * G.PISANI
3  REM * PROG. 5 - MEM-VHP
4  REM *
40  DIM MS(128),VS(128),HS(128),P
    S(128)
50  HOME
100  FOR M = 1024 TO 3071
105 P = 1 + (M > 2047)
108  IF P = 2 THEN  POKE  - 16299
    ,0
110 G =  INT ((M - P * 1024) / 12
    8)
120 R = M - P * 1024 - 128 * G
130 V = G + 1 +  INT (R / 40) * 8

140 H = R + 1 -  INT (R / 40) * 4
    0
150  IF V < 25 THEN  VTAB V: HTAB
    H: GOSUB 300: PRINT  CHR$ (4
    8 + P);: GOTO 170
160 S = S + 1:MS(S) = M:VS(S) = V
    :HS(S) = H:PS(S) = P
170  NEXT M
175  FOR A = 1 TO 1000: NEXT
180  POKE  - 16300,0
190  FOR A = 1 TO 1000: NEXT
200  HOME
210  PRINT "NUM   MEM   VTAB   HT
    AB   PTAB": VTAB 3: POKE 34,
    2
220  FOR A = 0 TO 15
230  FOR S = 1 + A * 8 TO 8 + A *
    8
240  PRINT S;: HTAB 6: PRINT MS(S
    );"     ";VS(S);"       ";HS(S
    );"     ";PS(S)
250  NEXT S
260  PRINT : INPUT "PREMI RETURN
    ";A$: PRINT
270  NEXT A
280  POKE 34,0: HOME : END
300  REM * PTAB
310 MV =  PEEK (41)
320  POKE 41,MV + 4 * (P = 2 AND
    MV < 8) - 4 * (P = 1 AND MV >
    7)
330  RETURN
```

le locazioni 40 e 41 automaticamente. Il programma 6, chiamato Base, dimostra quanto sopra, stampando a video gli indirizzi base di ciascuna riga che potrete confrontare con quelli della ormai famosa tabellà iniziale. Lasciamo a voi il compito di esaminare Base e di verificare con altri programmini la legge di variazione di 40 e 41.

L'indirizzo della posizione corrente del cursore, punto di partenza per Print, Input e Get, è dato da

PEEK(41) ★ 256 + PEEK(40) + PEEK (36)

quindi se si incrementa 41 di 4, prima di una Print Input o Get, si passa dalla pagina di memoria 4, 5, 6 e 7 rispettivamente in 8, 9, 10 e 11 e, senza modificare 40 e 36, si ottiene l'indirizzo della nuova posizione del cursore in pagina 2.

Eravamo rimasti al Gosub 300 di linea 150 del programma Mem-Vhp: la routine si incarica di effettuare le necessarie modifiche della locazione 41, eseguendo entrambi i cambi di pagina secondo il seguente schema:

- in Pagina 1 Ptab 1 non ha effetto;
- in Pagina 1 Ptab 2 cambia pagina (incrementa di 4 loc. 41);
- in Pagina 2 Ptab 1 cambia pagina (decrementa di 4 loc. 41);
- in Pagina 2 Ptab 2 non ha effetto.

Anche qui il programma alla fine dà i 16 gruppi di locazioni escluse con coordinate video illegali (linee 200-280).

Le istruzioni di I/O usate in pagina 2 presentano spesso un comportamento diverso da quello usuale. Una Print sulle righe 23 e 24, senza il ";" finale, non provoca lo scrool in pagina 2 ma lo può provocare in pagina 1. Due Print successive di cui la prima in pagina 2 provocano l'emissione della seconda in pagina 1: dopo la prima il sistema aggiorna automaticamente la locazione 37 (Vtab) e di conseguenza 40 e 41, la seconda Print, quindi, si troverà a lavorare con il cursore posizionato in pagina 1, a meno di non eseguire la famosa Ptab 2 tra le due Print. Una Print a "cavallo" di due righe scrive metà stringa in pagina 2 e l'altra metà in pagina 1. Questi ed altri esempi sono raccolti nel programma 7, chiamato Prove1-2, che vi dimostra l'uso in pagina 2 dei comandi di I/O (se volete allungare il tempo di pausa tra un'azione e l'altra per meglio seguire la dinamica tra causa ed effetto, aumentate le stesse routine (300 Ptab, 400 visualizzazione video, 500 posizionamento cursore, 600 lettura posizione cursore e 700 stampa posizione cursore con ";" finale).

Il programma è abbastanza semplice, ma introduce qualcosa di nuovo: le linee 11 e 16 ci risolvono in modo rapido e brillante un problema nuovo e ci aprono la

Apple: pagine video e dintorni

Listato 6. *Stampa sul video gli indirizzi base di ciascuna linea.*

```
1  REM *
2  REM * G. PISANI
3  REM * PROG. 6 - BASE
4  REM *
100  HOME
110  FOR V = 1 TO 24
120  VTAB V: HTAB V + 5
130 A =  PEEK (40):B =  PEEK (41)
   :H =  PEEK (36)
140  VTAB 23: HTAB 1: CALL  - 958

150  PRINT V;: HTAB 5: PRINT "40
   = ";A;
160  HTAB 15: PRINT "41 = ";B;"
   ";"MB = ";B * 256 + A;
170  PRINT "  ";"H = ";H: PRINT
180  INPUT "PREMI RETURN ";A$
190  NEXT V
200  END
```

Listato 7. *Il programma dimostra l'uso in pagina 2 dei comandi d'input/output.*

```
1  REM *
2  REM * G. PISANI
3  REM * PROG. 7 - PROVE1-2
4  REM *
5 A1 = 4000
6  POKE 33,40
10  HOME
11  PRINT  CHR$ (4);"BSAVE HOME,A
   1024,L1024"
12 P = 1:V = 1:H = 22: GOSUB 500
13  GOSUB 700: PRINT "SCROL"
14  FOR A = 1 TO A1: NEXT A
15  POKE  - 16299,0
16  PRINT  CHR$ (4);"BLOAD HOME,A
   2048"
20 P = 2:V = 1:H = 4: GOSUB 500
21  GOSUB 700: PRINT
22  GOSUB 600: GOSUB 700: PRINT "
   SCROL"
23  GOSUB 400
30 P = 2:V = 3:H = 25: GOSUB 500
31  GOSUB 700: PRINT "CAVALLO"
32  GOSUB 400
40 P = 2:V = 5:H = 1: GOSUB 500
41  GOSUB 700: INPUT "INPUT ? ";A
   $
45  GOSUB 300: GOSUB 600
46  GOSUB 700: PRINT "GET ";: GET
   A$
60 P = 2:V = 10:H = 5: GOSUB 500
61  GOSUB 700: PRINT "  S";
62  VTAB 11: GOSUB 300: PRINT "C"
   ;
63  VTAB 12: GOSUB 300: PRINT "A"
   ;
64  VTAB 13: GOSUB 300: PRINT "L"
   ;
65  VTAB 14: GOSUB 300: PRINT "A"
   ;
70 P = 2:V = 23:H = 1: GOSUB 500
71  GOSUB 700: PRINT "1 E 2 NO SC
   ROL"
72  GOSUB 400
80 P = 2: GOSUB 300
81  GOSUB 600: GOSUB 700: PRINT "
   1 SCROL E 2 NO"
82  GOSUB 400
100  POKE  - 16300,0: END
300  REM * PTAB
310 MV =  PEEK (41)
320  POKE 41,MV + 4 * (P = 2 AND
   MV < 8) - 4 * (P = 1 AND MV >
   7)
330  RETURN
400  REM * ASPETTA E CAMBIA 2-1-2

410  FOR A = 1 TO A1: NEXT A: POKE
    - 16300,0
420  FOR A = 1 TO A1: NEXT A: POKE
    - 16299,0: RETURN
500  REM * POSIZIONA IL CURSORE
510  VTAB V: HTAB H: GOSUB 300
520  RETURN
600  REM * RILEVA LA POSIZIONE
610 P = 1 + ( PEEK (41) > 7)
620 V =  PEEK (37) + 1
630 H =  PEEK (36) + 1
640  RETURN
700  REM * STAMPA LA POSIZIONE
710  PRINT "P=";P;" V=";V;" H=";H
   ;"  ";
720  RETURN
```

strada verso una nuova tecnica di gestione del video inteso come entità globale. Il programma pratico: manca una istruzione di Home valida per la pagina 2, dovremmo utilizzare una routine che stampa 24 stringhe di 40 caratteri bianchi in pagina 2 dopo aver proceduto per ogni riga ad eseguire il solito Gosub 300, metodo lento e pesante. La soluzione: la linea 11, una Bsave, che preleva la memoria da 1024 per lunghezza 1024, crea un File binario chiamato Home che è la "fotografia" del nostro video ripulito da una precedente istruzione di Home. Ora la linea 16 rimette lo stesso file in memoria a partire dall'indirizzo 2048, cioè riempie la pagina 2 in blocco con il codice 160 che rappresenta il carattere bianco. Tale metodo ci permette di conservare videate, maschere già pronte, di cambiare velocemente il video 2 mentre l'utente sta leggendo il video 1, di aggiornare tabelle senza doverle riscrivere ecc. Si perde un po' di spazio sul dischetto (mica tanto poi), ma si guadagna in velocità ed efficienza.

Attenzione, però, se non volete avere rogne, ricordatevi di quanto detto a proposito delle locazioni speciali della pagina 1 (vi avevamo promesso un esempio, o no?): se salvate, per esempio, una videata dal drive 1 slot 4 e la riutilizzate inserendo il dischetto nel drive 2 slot 6, avrete la sgradita sorpresa di accorgervi che il primo comando DOS non indirizzato si rivolgerà allo slot 4 invece che al 6. Ricaricando la videata abbiamo sostituito i valori delle locazioni speciali con quelli presenti al momento del salvataggio che si riferiscono a configurazione diversa: prelevate con Peek la o le locazioni che interessano prima del Bload e rimettetele poi al loro posto con Poke (almeno la locazione 2040 che contiene lo slot in uso).

Proponiamo alcuni utili esercizi per applicare quanto detto:

- provate ad utilizzare le finestre (locazioni 32, 33, 34 e 35) con la pagina 2;
- scoprirete che la cosa è ancora possibile;
- provate a scrivere un algoritmo in BASIC o in Assembly che sostituisca, per la pagina 2, il Call -868 (pulizia della riga dal cursore in poi);
- come sopra per Call -958 (pulizia del video dal cursore in poi);
- realizzate routine in Assembly per implementare Ptab e Home2;
- come sopra per liberare l'area del video 2.

Gioco d'inseguimento riscritto con Peek e Poke

Il gioco d'inseguimento, nella versione di Poma, usava le Peek (linee 184-187) per "leggere" il muro, e Vtab, Htab e Print (linee 100, 170 e 190) per "scrivere" le posizioni dei contendenti. Questa operazione di "scrittura" può essere realizzata con Poke riferite a Q2, locazione di memoria del giocatore, e MC, per il computer.

Il listato 8, chiamato Lab-Po/V0 (da labirinto), contiene le modifiche da apportare al programma di Poma per ottenere la nuova versione Lab-V0. La linea 97 inizializza i valori di Q2 ed MC usando rispettivamente gli algoritmi di conversione 500 e 600. Le linee 182-183 ed i test 500-530 sono scomparsi perché facenti parte del vecchio algoritmo. Suggeriamo di cambiare i tasti di comando con I, M, J e K che sono più usuali per l'Apple essendo già usati per le funzioni di Escape (120-135 e 184-187). In tal caso i codici dei tasti diventano nell'ordine 203, 202, 205, 201. Una rapida prova dimostra che ora il programma gira molto veloce, l'istruzione 95 non è più sufficente neanche con Speed = 1 (i simboli in movimento quasi non si vedono). Tolte quindi 7 e 95 abbiamo aggiunto un loop di attesa in linea 205 con variabile inizializzata in linea 7.

Se provate a giocare molto lentamente (VE=500) vi accorgerete che:

- 120-135 permettono al giocatore di "rifugiarsi" sul muro esterno (X e Y sono prese valide prima di andare a "vedere" il muro);
- 150-170 col giocatore sul muro il Computer può raggiungerlo e romperlo;
- 184-187 il giocatore si accorge di esser sul muro e ritorna indietro;
- col muro ormai aperto è possibile per il giocatore

Listato 8. *Modifiche per il listato 8 bis.*

```
0   REM *
1   REM * G.PISANI
2   REM * LISTATO 8 - LAB-PO/VO
3   REM * MODIFICHE GIOCO D'INSEGU
      IMENTO DA VERSIONE POMA A LA
      B-VO
4   REM *
7 VE = 100: REM * SPEED= 255
95  REM * SPEED= 150
97  GOSUB 500: GOSUB 600
100  POKE Q2,160: POKE MC,160
170  GOSUB 600: POKE MC,216
182  DEL
183  DEL
190  GOSUB 500: POKE MG,170
205  FOR T = 1 TO VE: NEXT
500 Q2 = 895 + 128 * X - 984 *  INT
     ((X - 1) / 8) + Y: RETURN
510  DEL
520  DEL
530  DEL
600 MC = 895 + 128 * A - 984 *  INT
     ((A - 1) / 8) + B: RETURN
```

Apple: pagine video e dintorni

Listato 8 bis. *Il programma Gioco di inseguimento.*

```
0   REM *
1   REM * C.POMA E G.PISANI
2   REM * PROG. 8 - LAB-VO
3   REM * GIOCO D'INSEGUIMENTO IN
      VERSIONE POMA MODIFICATO
4   REM *
5   HOME
7 VE = 100: REM * SPEED= 255
10   PRINT "##################"
20   FOR C = 1 TO 8
30   PRINT "#     #     #     #"
40   NEXT
50   PRINT "##################"
55   VTAB 3: HTAB 25: PRINT "RECOR
      D : ";RE
60 A = 7:B = 14:X = 2:Y = 2:P = 0

95   REM * SPEED= 150
97   GOSUB 500: GOSUB 600
100  POKE Q2,160: POKE MC,160
110 K =  PEEK ( - 16384)
111 A$ =  CHR$ (K)
120  IF  ASC (A$) = 203 THEN Y =
      Y + 1
125  IF  ASC (A$) = 202 THEN Y =
      Y - 1
130  IF  ASC (A$) = 205 THEN X =
      X + 1
135  IF  ASC (A$) = 201 THEN X =
      X - 1
140 H =  INT ( RND (1) * 2)
150  IF H = 0 THEN A = A + (X > A
     ) - (X < A)
160  IF H = 1 THEN B = B + (Y > B
     ) - (Y < B)
170  GOSUB 600: POKE MC,216
180  IF A = X AND B = Y THEN 220
181 NN = 163: GOSUB 500
184  IF  PEEK (Q2) = NN AND  ASC
     (A$) = 203 THEN Y = Y - 1
185  IF  PEEK (Q2) = NN AND  ASC
     (A$) = 202 THEN Y = Y + 1
186  IF  PEEK (Q2) = NN AND  ASC
     (A$) = 205 THEN X = X - 1
187  IF  PEEK (Q2) = NN AND  A6C
     (A$) = 201 THEN X = X + 1
190  GOSUB 500: POKE Q2,170
200 P = P + 1
205  FOR T = 1 TO VE: NEXT
210  GOTO 100
220  VTAB 15: PRINT "PUNTI : ";P
225  IF P > RE THEN RE = P
230  PRINT : PRINT "PER CONTINUAR
     E PREMI <RETURN> ";: INPUT Q
     $
240  GOTO 5
500 Q2 = 895 + 128 * X - 984 *  INT
     ((X - 1) / 8) + Y: RETURN
600 MC = 895 + 128 * A - 984 *  INT
     ((A - 1) / 8) + B: RETURN
```

(bravi se ci riuscite col gioco veloce) uscire dal recinto attraverso il varco. In tal caso, infatti, i test 184-187 non hanno più efficacia;

- il giocatore può quindi arrivare in posizione, poniamo, X = 25, con stop del programma in versione Poma (Htab 25 genera un Error) o con una Poke a memorie fuori video con programma in versione Lab-V0 (con conseguenze incalcolabili).

Per eliminare questo difetto abbiamo riscritto il programma in versione Lab-V1 che corregge anche qualche altro inconveniente della vecchia versione e introduce l'uso di alcune delle tecniche di gestione del video di cui si è parlato. Non è stato possibili mantenere la stessa numerazione delle precedenti versioni pubblicate per ragioni di spazio.

REMarks

120, 130 e **150** - Calcolano le locazioni MP, MR, MA, MQ ed MV, tramite la routine 1000, in funzione delle coordinate A e B fornite. Saranno usate per scrivere e leggere rispettivamente il punteggio, il record ed il record assoluto (tutti sulla riga 17), e le due richieste del rigo 19 (vedi linea 140). La linea 130, inoltre, azzera il record sul video (Poke MR, 176);

200-240 e **900** - Si incaricano di gestire il record assoluto RR direttamente a video senza memorizzarlo in un file tradizionale. 900 salva 3 byte (MA-2, MA-1, MA) del video (il record) in un file binario (V1-Rec) che viene rimesso a video da 210 e decodificato in RR dal loop 220-240 (che scarta anche i caratteri bianchi). La prima volta il file V1-Rec non c'è, ma l'errore è ripreso da 200 che evita problemi saltando anche la decodifica.

Una simile soluzione, forse esagerata, serve a chiarire la potenzialità della gestione video con Bload e Bsave.

250 - Ristabilisce il normale uso dell'errore e chiude la finestra per salvare quanto scritto da successivi Home.

300-350 - Sono le vecchie 5-50 che preparano il recinto nuovo.

360 - Azzera il punteggio sul video all'inizio di ogni partita (sempre con Poke diretti).

Listato 9. *Il programma Lab-V1.*

```
1    REM *
2    REM * G. PISANI
3    REM * PROG. 9 - LAB-V1
4    REM * GIOCO D'INSEGUIMENTO AD
       1 PIANO
5    REM *
100  HOME : VTAB 14: HTAB 35: PRINT
     "RECORD";
110  PRINT "PUNTI"; SPC( 11);"REC
     ORD"; SPC( 10);"ASSOLUTO"
120 A = 17:B = 4: GOSUB 1000:MP =
     MC
130 B = 21: GOSUB 1000:MR = MC: POKE
     MR,176:B = 38: GOSUB 1000:MA
     = MC
140  VTAB 19: PRINT "CONTINUI (S/
     N) ?      VELOCITA' (0-9) ? "

150 A = 19:B = 18: GOSUB 1000:MQ =
     MC:B = 40: GOSUB 1000:MV = M
     C
200  ONERR  GOTO 250
210  PRINT  CHR$ (4);"BLOAD V1-RE
     C"
220  FOR MJ = 0 TO 2
230 CF =  PEEK (MA - MJ) - 176:RR
     = RR + CF * (10 ^ MJ) * (CF
     >  = 0)
240  NEXT
250  POKE 216,0: POKE 35,13
300  HOME
310  PRINT "#################"
320  FOR C = 1 TO 8
330  PRINT "#     #     #     #"
340  NEXT
350  PRINT "#################"
360 P = 0: POKE MP,176: POKE MP -
     1,160: POKE MP - 2,160
400 A = 9:B = 15: GOSUB 1000: POKE
     MC,216
410 X1 = 2:Y1 = 2: GOSUB 1100:X =
     X1:Y = Y1:MG = MM: POKE MG,1
     70
500  POKE MV,121 - VE
510 K =  PEEK ( - 16384): IF (K <
     176 OR K > 185) AND K <  > 1
     41 GOTO 510
520  IF K = 141 THEN K = 185 - VE

530  POKE MV,K:VE = 185 - K
540  FOR Q = 1 TO 1000: NEXT :W =
      PEEK ( - 16368): PRINT  CHR$
     (7)
600 H =  INT ( RND (1) * 2)
610  IF H = 0 THEN A = A + (X > A
     ) - (X < A)
620  IF H = 1 THEN B = B + (Y > B
     ) - (Y < B)
640 CU = MC: GOSUB 1000: IF CU =
     MC GOTO 700
660  POKE CU,160: POKE MC,216: IF
     MC = MG GOTO 800
670  POKE CU,160: POKE MC,216: IF
     MC = MG GOTO 800
700  FOR T = 1 TO VE * 50: NEXT :
     K =  PEEK ( - 16384):W =  PEEK
     ( - 16368)
710  IF K < 201 OR K > 205 OR K =
     204 GOTO 600
720 X1 = X + (K = 205) - (K = 201
     )
730 Y1 = Y + (K = 203) - (K = 202
     )
750  GOSUB 1100: IF  PEEK (MM) =
     163 GOTO 600
780  POKE MG,160: IF MM = MC GOTO
     800
790 X = X1:Y = Y1:MG = MM: POKE M
     G,170: GOSUB 1200: GOTO 600
800  IF P > R THEN LS = MR:LR = 3
     : GOSUB 1400
810  IF P > RR THEN LS = MA:LR =
     3: GOSUB 1400
820  POKE MQ,96
830 K =  PEEK ( - 16384): IF K <
      > 206 AND K <  > 211 GOTO 8
     30
840  POKE MQ,K:W =  PEEK ( - 1636
     8)
860  IF P > R THEN R = P:LS = MR:
      GOSUB 1300
870  IF P > RR THEN RR = P:LS = M
     A: GOSUB 1300
880  IF K = 211 GOTO 300
900  PRINT  CHR$ (4);"BSAVE V1-RE
     C,A";MA - 2;",L";3
910  POKE 35,24: VTAB 20: END
1000 MC = 895 + 128 * A - 984 *  INT
     ((A - 1) / 8) + B: RETURN
1100 MM = 895 + 128 * X1 - 984 *
     INT ((X1 - 1) / 8) + Y1: RETURN
```

Apple: pagine video e dintorni

Seguito listato 9.

```
1200 P = P + 1
1210  FOR MJ = 0 TO 2
1220 CF =  PEEK (MP - MJ): IF CF =
     160 THEN CF = 176
1240  POKE MP - MJ,CF + 1 - 10 *
     (CF = 185): IF CF < 185 GOTO
     1260
1250  NEXT
1260  RETURN
1300  FOR MJ = 0 TO 2: POKE LS -
   MJ, PEEK (MP - MJ): NEXT : RETURN

1400  FOR L = 0 TO LR - 1
1410 CF =  PEEK (LS - L) - 64:CF =
     CF - 64 * (CF > 127): POKE L
     S - L,CF
1420  NEXT
1430  RETURN
```

400 - Fissa le coordinate iniziali (A e B) e la locazione di memoria (MC) del Computer (routine 1000) e scrive a video la posizione di partenza (Poke MC).
410 - Inizializza la posizione del giocatore (X, Y ed MG) usando l'algoritmo di linea 1100 che ci dà MM in funzione di X1 ed Y1 che, come vedremo oltre, sono la posizione ipotetica del giocatore.
500-530 - Richiede a video la velocità del gioco. Ne parleremo più avanti.
540 - Dopo un breve loop di ritardo azzera la tastiera (Peek -16368) ed emette un beep di avviso inizio gara (Print Chr$(7)).
600-620 - Sono le vecchie 140-160 che muovono Random il Computer.
640-670 - Implementano, se CU (vecchio) < > MC (nuovo), il movimento della X in modo molto costante (la sua comparsa, ex 170, segue subito la sua scomparsa, ex 100). 670 controlla se la X ha raggiunto il suo obiettivo.
700 - È il loop di ritardo che dà la velocità del gioco: VE varia da 0 a 9 dove il numero maggiore rappresenta un maggior ritardo ed una minor velocità, motivo per cui i valori richiesti a video (indici di velocità) sono opposti a VE. Dopo il loop c'è la lettura da tastiera (ex 110) della mossa decisa dal giocatore ed il suo successivo azzeramento. Nella precedente versione la pressione di un solo tasto provocava la continua rilettura del suo codice e l'* si spostava senza sosta fino al primo muro o fino ad un cambio di direzione ottenuto con nuovo tasto. La Peek -16368 abbassa il valore della locazione dove è contenuto l'ultimo tasto premuto al di sotto di 128. Senza altri tasti K verrà letto < 128.
710 - Evita qualunque operazione se K non corrisponde ad uno dei quattro tasti leciti (I, J, K ed M).
720-750 - (ex 120-135) Calcolano le nuove coordinate ipotetiche di * (X1 ed Y1) e la relativa locazione (MM): se siamo sul muro (If di 750) abbiamo sempre validi X, Y ed MG e non occorrono spostamenti di *.
780 - Cancella l'* dalla vecchia posizione MG e si incarica di controllare se con la mossa fatta il giocatore finisce tra le braccia del suo inseguitore.
790 - Assume valida la posizione ipotetica del giocatore (X1, Y1 ed MM diventano X, Y ed MG) e ripiazza l'*. Anche qui il movimento appare costante (il tempo tra la comparsa di *, ex 190, e la sua scomparsa risultava minore di quello inverso). Quando il giocatore ha effettuato una mossa valida viene aggiornato il punteggio (routine 1200). Prima il punteggio veniva aumentato comunque, anche senza alcun intervento sulla tastiera.
800-810 - Rendono lampeggiante il record eventualmente battuto (routine 1400).
820-840 - Simula un Get a video completo di lampeggio del cursore. Accetta solo S o N. Ripulisce la tastiera e provvede a scrivere a video la risposta letta.

Apple: pagine video e dintorni

860-870 - Aggiornano a video i record battuti (routine 1300).
880 - test sulla risposta del Get, se è S il gioco continua.
900-910 - Chiude il gioco riaprendo la finestra e salvando il record.
500-530 - Richiede la velocità a video: il cursore si sovrappone al valore esistente senza cancellarlo, le risposte sono controllate (Return conferma il valore vecchio) e scritte a video. Dei valori di VE si è già parlato. La simulazione del Get differisce alquanto da quella delle linee 820-840.
1000-1100 - Sono gli algoritmi di conversione per MC ed MM.
1200-1260 - Realizza un contatore a video facendo scattare le cifre come quelle di un orologio (sempre con i soliti Poke).
1300 - Copia parti di video (lunghe 3) da un'area (che termina in MP) ad un'altra (che termina in LS).
1400-1430 - È un simpatico algoritmo che fa lampeggiare un campo scritto in Normal, lungo LR che termina in LS. Provate ad ampliarlo in modo che converta anche i caratteri Inverse, ignorando i caratteri già Flash. Scrivete poi gli algoritmi per convertire un intero campo in Inverse oppure in Normal.
Possiamo ora pensare di realizzare un recinto spaziale a due piani sovrapposti: quello superiore nella pagina video 1 e quello inferiore nella pagina video 2. Se gestiamo il gioco in modo da inquadrare sempre il piano dove si muove il nostro giocatore (per il quale useremo 1 e 2,

Listato 10. *Modifiche al Gioco d'inseguimento da versione Lab-V1 a Lab-V2.*

```
1    REM *
2    REM * G. PISANI
3    REM * LISTATO 10 - LAB-V1/V2
4    REM * MODIFICHE GIOCO D' INSEGU
        IMENTO DA VERSIONE LAB-V1 A
        LAB-V2
5    REM *
10 AB = 2.5
120 A = 17:B = 4:C = 1: GOSUB 100
        0:MP = MC
210   PRINT  CHR$ (4);"BLOAD V2-RE
        C"
310   PRINT "################"; SPC(
        10);"PAG. 1"
370   PRINT  CHR$ (4);"BSAVE V2-VI
        DEO,A1024,L1024"
380   PRINT  CHR$ (4);"BLOAD V2-VI
        DEO,A2048"
390   VTAB 1: HTAB 32: POKE 41, PEEK
        (41) + 4: PRINT "2"
400 A = 9:B = 15:C = 2: GOSUB 100
        0: POKE MC,216: GOSUB 1500: POKE
        MW,CH
410 X1 = 2:Y1 = 2:Z1 = 1: GOSUB 1
        100:X = X1:Y = Y1:Z = Z1:MG =
        MM: POKE MG,177
530   POKE MV,K: POKE MV + 1024,K:
        VE = 185 - K
600 H =  INT ( RND (1) * AB)
630   IF H = 2 THEN C = Z
650   POKE MW,PU: GOSUB 1500: IF M
        C < > MG + 1024 THEN  POKE
        MW,CH
660 PC = 160: IF CU = MW THEN PC =
        CH
670   POKE CU,PC: POKE MC,216: IF
        MC = MG GOTO 800
710   IF K < 201 OR K > 205 GOTO 6
        00
740 Z1 = Z: IF K = 204 THEN Z1 =
        3 - Z
760   IF K = 204 THEN Z = Z1: POKE
        (Z - 16301),0
770 PG = 160: IF MG = MW THEN PG =
        CH
780   POKE MG,PG: IF MM = MC GOTO
        800
785 CG = 176 + Z: IF MM = MC + 10
        24 THEN CG = CH
790 X = X1:Y = Y1:MG = MM: POKE M
        G,CG: GOSUB 1200: GOTO 600
800   IF P > R THEN LS = MR + 1024
        * (Z = 2):LR = 3: GOSUB 140
        0
810   IF P > RR THEN LS = MA + 102
        4 * (Z = 2):LR = 3: GOSUB 14
        00
820   POKE MQ + 1024 * (Z = 2),96
840   POKE MQ,K: POKE MQ + 1024,K:
        W =  PEEK ( - 16368)
850   POKE  - 16300,0
900   PRINT  CHR$ (4);"BSAVE V2-RE
        C,A";MA - 2;",L";3
1000 MC = C * 1024 - 129 + 128 *
        A - 984 *  INT ((A - 1) / 8)
        + B: RETURN
1100 MM = Z1 * 1024 - 129 + 128 *
        X1 - 984 *  INT ((X1 - 1) /
        8) + Y1: RETURN
1230  POKE MP + 1024 - MJ,CF + 1 -
        10 * (CF = 185)
1500 MW = MC + 1024 * (C = 1) - 1
        024 * (C = 2)
1510 CH = 88 - 64 * (C = 1)
1520 PU = 160 + 3 * ( PEEK (MW) =
        163)
1530  RETURN
```

anziché *, a seconda di dove si trova), ci troveremo di fronte a tre casi possibili:
A) giocatore e Computer sono sullo stesso piano: non si presentano problemi;
B) giocatore sul piano superiore (pag. 1), Computer su quello inferiore (pag. 2): allora in pagina 1 (inquadrata) si vedrà solo l'immagine del Computer che si muove in profondità, immagine che rappreseteremo con una X lampeggiante (codice 88). Il Computer in profondità può abbattere i muri del recinto di pagina 2, mentre la sua proiezione in superfice li attraversa solo. La X lampeggiante verrà coperta dall'1 del giocatore se questi passa sopra il Computer;
C) giocatore sul piano inferiore (pag. 2), Computer su quello superiore (pag. 1): nella pagina 2 (inquadrata) si vedrà solo l'ombra del Computer che si muove più in alto, ombra che rappresenteremo con una X Inverse (codice 24). Anche qui la proiezione attraversa i muri di pagina 2 senza romperli, mentre il Computer può rompere quelli di pagina 1. Se il Computer sorvolerà il giocatore, la sua ombra sinistra ne coprirà il 2.
Per comandare il passaggio di piano, il giocatore ha a disposizione il tasto L che, oltre ad essere vicino agli altri tasti di comando, ha come codice ASCII 204 il che semplificherà i test sulla validità della mossa. Il listato 10, chiamato Lab-V1/V2, contiene tutte le modifiche da apportare a Lab-V1 per ottenere la nuova versione sopra descritta.
Di seguito troverete l'analisi delle differenze (per gli impazienti che volessero provare subito il programma ricordiamo di far girare prima il programma Video 2 per liberare la pagina 2 dai suoi abituali compiti):
1000 e **1100** - Sono i nuovi algoritmi di conversione da coordinate (A, B e C per Computer e X1, Y1 e Z1 per giocatore) a locazione di memoria (MC ed MM).
120 - Pone C=1 per calcolare correttamente MP, MR, MA, MQ ed MV tramite 1000.
210 e **900** - Il nome del file binario per il record è ora V2-Rec (i record dei due casi sono non omogenei e devono rimanere separati).
310 - Scrive "Pag. 1" in pagina 1 come identificatore.
370-380 - Preparano il recinto inferiore in pagina 2: il modo più semplice è quello di prelevare tutta la pagina 1 con Bsave e ricopiarla in pagina 2 con Bload. Abbiamo così ricopiato anche tutti i record e tutto il resto.
390 - Completa in pagina 2 l'identificatore "Pag. 2" sostituendo "1" (copiato con la videata) con "2", mediante l'uso di un Print e dallo spiazzamento di 4 settori dell'indirizzo iniziale della riga di stampa (Poke 41), tecnica di cui si è già parlato.
400 - Posiziona il Computer nell'angolo opposto al giocatore (C=2), mentre pone nel piano superiore la sua proiezione lampeggiante (la routine 1500 fornirà la sua posizione MW ed il suo codice CH).
410 - Pone Z1 e Z pari a 1 per la posizione iniziale del

Listato 10 bis. *Il Gioco d'inseguimento a due piani.*

```
1   REM *
2   REM * G. PISANI
3   REM * PROG. 10 - LAB-V2
4   REM * GIOCO D'INSEGUIMENTO A 2
       PIANI
5   REM *
10 AB = 2.5
100  HOME : VTAB 14: HTAB 35: PRINT
     "RECORD";
110  PRINT "PUNTI"; SPC( 11);"REC
     ORD"; SPC( 10);"ASSOLUTO"
120 A = 17:B = 4:C = 1: GOSUB 100
     0:MP = MC
130 B = 21: GOSUB 1000:MR = MC: POKE
     MR,176:B = 38: GOSUB 1000:MA
      = MC
140  VTAB 19: PRINT "CONTINUI (S/
     N) ?      VELOCITA' (0-9) ? "

150 A = 19:B = 18: GOSUB 1000:MQ =
     MC:B = 40: GOSUB 1000:MV = M
     C
200  ONERR  GOTO 250
210  PRINT  CHR$ (4);"BLOAD V2-RE
     C"
220  FOR MJ = 0 TO 2
230 CF =  PEEK (MA - MJ) - 176:RR
      = RR + CF * (10 ^ MJ) * (CF
      )  = 0)
240  NEXT
250  POKE 216,0: POKE 35,13
300  HOME
310  PRINT "################"; SPC(
     10);"PAG. 1"
320  FOR C = 1 TO 8
330  PRINT "#     #     #     #"
340  NEXT
350  PRINT "################"
360 P = 0: POKE MP,176: POKE MP -
     1,160: POKE MP - 2,160
370  PRINT  CHR$ (4);"BSAVE V2-VI
     DEO,A1024,L1024"
380  PRINT  CHR$ (4);"BLOAD V2-VI
     DEO,A2048"
390  VTAB 1: HTAB 32: POKE 41, PEEK
      (41) + 4: PRINT "2"
400 A = 9:B = 15:C = 2: GOSUB 100
     0: POKE MC,216: GOSUB 1500: POKE
     MW,CH
```

PERSONAL SOFTWARE

Apple: pagine video e dintorni

Seguito listato 10 bis.

```
410 X1 = 2:Y1 = 2:Z1 = 1: GOSUB 1
     100:X = X1:Y = Y1:Z = Z1:MG =
     MM: POKE MG,177
500  POKE MV,121 - VE
510 K =  PEEK ( - 16384): IF (K <
     176 OR K > 185) AND K < > 1
     41 GOTO 510
520  IF K = 141 THEN K = 185 - VE

530  POKE MV,K: POKE MV + 1024,K:
     VE = 185 - K
540  FOR Q = 1 TO 1000: NEXT :W =
      PEEK ( - 16368): PRINT  CHR$
     (7)
600 H =  INT ( RND (1) * AB)
610  IF H = 0 THEN A = A + (X > A
     ) - (X < A)
620  IF H = 1 THEN B = B + (Y > B
     ) - (Y < B)
630  IF H = 2 THEN C = Z
640 CU = MC: GOSUB 1000: IF CU =
     MC GOTO 700
650  POKE MW,PU: GOSUB 1500: IF M
     C < > MG + 1024 THEN  POKE
     MW,CH
660 PC = 160: IF CU = MW THEN PC =
     CH
670  POKE CU,PC: POKE MC,216: IF
     MC = MG GOTO 800
700  FOR T = 1 TO VE * 50: NEXT :
     K =  PEEK ( - 16384):W =  PEEK
     ( - 16368)
710  IF K < 201 OR K > 205 GOTO 6
     00
720 X1 = X + (K = 205) - (K = 201
     )
730 Y1 = Y + (K = 203) - (K = 202
     )
740 Z1 = Z: IF K = 204 THEN Z1 =
     3 - Z
750  GOSUB 1100: IF  PEEK (MM) =
     163 GOTO 600
760  IF K = 204 THEN Z = Z1: POKE
     (Z - 16301),0
770 PG = 160: IF MG = MW THEN PG =
     CH
780  POKE MG,PG: IF MM = MC GOTO
     800
785 CG = 176 + Z: IF MM = MC + 10
     24 THEN CG = CH
790 X = X1:Y = Y1:MG = MM: POKE M
     G,CG: GOSUB 1200: GOTO 600
800  IF P > R THEN LS = MR + 1024
      * (Z = 2):LR = 3: GOSUB 140
     0
810  IF P > RR THEN LS = MA + 102
     4 * (Z = 2):LR = 3: GOSUB 14
     00
820  POKE MQ + 1024 * (Z = 2),96
830 K =  PEEK ( - 16384): IF K <
      > 206 AND K < > 211 GOTO 8
     30
840  POKE MQ,K: POKE MQ + 1024,K:
     W =  PEEK ( - 16368)
850  POKE  - 16300,0
860  IF P > R THEN R = P:LS = MR:
      GOSUB 1300
870  IF P > RR THEN RR = P:LS = M
     A: GOSUB 1300
880  IF K = 211 GOTO 300
900  PRINT  CHR$ (4);"BSAVE V2-RE
     C,A";MA - 2;",L";3
910  POKE 35,24: VTAB 20: END
1000 MC = C * 1024 - 129 + 128 *
     A - 984 *  INT ((A - 1) / 8)
      + B: RETURN
1100 MM = Z1 * 1024 - 129 + 128 *
     X1 - 984 *  INT ((X1 - 1) /
     8) + Y1: RETURN
1200 P = P + 1
1210  FOR MJ = 0 TO 2
1220 CF =  PEEK (MP - MJ): IF CF =
     160 THEN CF = 176
1230  POKE MP + 1024 - MJ,CF + 1 -
     10 * (CF = 185)
1240  POKE MP - MJ,CF + 1 - 10 *
     (CF = 185): IF CF < 185 GOTO
     1260
1250  NEXT
1260  RETURN
1300  FOR MJ = 0 TO 2: POKE LS -
   MJ, PEEK (MP - MJ): NEXT : RETURN

1400  FOR L = 0 TO LR - 1
1410 CF =  PEEK (LS - L) - 64:CF =
     CF - 64 * (CF > 127): POKE L
     S - L,CF
1420  NEXT
1430  RETURN
1500 MW = MC + 1024 * (C = 1) - 1
     024 * (C = 2)
1510 CH = 88 - 64 * (C = 1)
1520 PU = 160 + 3 * ( PEEK (MW) =
     163)
1530  RETURN
```

Apple: pagine video e dintorni

giocatore.
530 - Una volta letta la velocità voluta (operazione che, come vedremo, avviene sempre in pagina 1) aggiorna il valore anche in pagina 2 (MV + 1024).
10/60 e **630** - Il Computer deve ora muoversi Random su tre dimensioni: ponendo AB pari a 3 otterremo per H i valori 0, 1 e 2 ma con eguale probabilità.
Ora usando un AB compreso tra 2 e 3 la probabilità di avere H=2 diminuisce al diminuire di AB. Con H=2 il Computer si muoverà solo se il giocatore è su di un piano diverso, quindi AB diventa un indice di difficoltà nel gioco regolando la frequenza con cui il Computer segue il giocatore che ha cambiato piano. Un valore di compromesso mi sembra sia 2,5. La linea 630 adegua la coordinata C del Computer a quella Z del giocatore qualora H sia pari a 2.
650 - La mossa del Computer (linea 670 con If di 640 non verificato) comporta anche lo spostamento della sua proiezione: questa lasciando il vecchio MW per il nuovo deve ripristinare la condizione iniziale preesistente al suo passaggio. La citata routine 1500 deve quindi provvedere a conservare in PU (codice di pulitura) il carattere da ripristinare. Con l'If finale si evita di scrivere la proiezione nel caso in cui il Computer finisce sotto al giocatore, come già spiegato. Notare che 650 e 670 gestiscono lo spostamento del Computer e della sua proiezione a prescindere da ciò che inquadra il video (il Computer può abbattere muri anche se non visto).
660-670 - La prima prepara PC (carattere di pulitura per MC che si sposta, vedi 670): sarà generalmente blank (160) a meno che non ci sia un cambio di piano (un blank cancellerebbe allora la proiezione già scritta da 650), allora PC sarà pari a CH.
710 - È il nuovo test sul tasto usato (ora accetta anche L).
740 - Ricostruisce Z1 (dopo X1 ed Y1) che sarà pari a 3-Z se si digita la L.
760 - Cambia la pagina inquadrata e posiziona Z pari al provvisorio Z1.
770-780 - La prima prepara PG (carattere di pulitura per MG che si sposta, vedi 780): sarà generalmente blank (160) a meno che MG (giocatore in partenza) ed MW (proiezione Computer) non risultino eguali, allora PG sarà pari a CH.
785-790 - La prima prepara CG (carattere da scrivere nella nuova posizione del giocatore, vedi 790): sarà generalmente 1 o 2 a seconda del piano (177 e 178), a meno che la posizione non coincida con l'ombra del Computer, allora sarà pari a CH.
800-810 - Eseguono il lampeggio del record nella pagina che risulta inquadrata.
820, 840 e **850** - La simulazione del Get va fatta sulla pagina inquadrata (820), mentre la risposta va piazzata su entrambe le pagine (840) in quanto occorre tornare comunque in pagina 1 (850).
1230 - Aggiorna il punteggio anche in pagina 2.
1500-1530 - Calcola MW (posizione della nuova proiezione di MC), CH suo relativo codice e PU carattere necessario a ripulire il video dopo il passaggio (160 o 163, blank e muro usualmente). Notare che PU=Peek(MW) non sarebbe corretto in quanto la posizione della proiezione potrebbe coincidere sia con quella del giocatore che con quella del Computer (cambio pagina) non ancora cancellata, in tal caso si conserverebbe un codice errato. Abbiamo già visto come sono gestiti i casi particolari di sovrapposizione.

Conclusioni

Quanto detto finora non ha la pretesa di esaurire il panorama delle tecniche alternative per la gestione del video, ma solo di aprire un discorso.
Speriamo, infatti, di leggere quanto prima su queste stesse pagine, nuovi contributi sull'argomento, magari anche dell'amico Poma, il cui lavoro è stato per noi un prezioso spunto iniziale. Vorremmo al proposito lanciare un interessante quesito la cui soluzione potrebbe forse aprire nuove strade.
È evidente che l'Apple gestisce il video, le sue finestre, e tutti gli indirizzi interni di riferimento, di cui si è parlato, a partire da un unico indirizzo base, quello per intenderci che rappresenta l'angolo superiore sinistro del video (1024). Ora non siamo ancora riusciti a trovare le locazioni dove viene conservato questo valore: trovarlo, sempre che esista, significa poterlo modificare a nostro piacimento ricreando, forse, le condizioni standard per la pagina 1 anche in pagina 2.
Non sappiamo se esiste una risposta a tale quesito o se effettivamente possa portare a qualcosa di veramente positivo, ma speriamo che possa produrre almeno qualche minuto di riflessione in qualcuno dei tanti appledipendenti sparsi per l'Italia e chissà se ...
Un consiglio: se volete tornare a lavorare ad un video solo dopo aver fatto girare Video2 non vi basterà un Reset, o riaccendete la macchina o usate un programma Video 1: ma lo dovrete prima scrivere da soli. Buon divertimento. ■

COMMODORE VIC 20- C 64

Type Writer e dimensionamento dinamico

di Alessandro Guida

Questo mese pubblichiamo un interessante intervento di Maurizio Paolinelli - Prima però facciamo alcuni chiarimenti sul programma Type Writer per VIC20 pubblicato nei numeri 18 e 19 di **Personal Software**.

Poiché questo programma ha avuto un certo successo e continua ad essere molto richiesto, abbiamo cercato di eliminare ogni possibile causa di malfunzionamento. Si sono riscontrate due imprecisioni nel testo dell'articolo e due errori nel programma.
Sicuri di fare cosa gradita a chi usa questo programma, riportiamo le correzioni necessarie.

Imprecisioni nel testo

1 - Nel testo si parla della gestione di piccoli file sequenziali. Questo non è in effetti possibile col programma pubblicato sulla rivista poiché i testi vengono registrati su disco col suffisso Prg e non Seq.
2 - Il simbolo per riportare il risultato di un calcolo nel testo è sbagliato. Va usata la freccia in alto ↑ e non il punto esclamativo, come appare nell'articolo.

Errori nel programma

1 - Nell'uso del registratore per il Load o Save di testi il programma risulta bloccarsi dopo aver completato l'operazione.
Per evitare ciò, è necessario digitare le seguenti righe:

```
3890 SYS(23135):IFR$="F"THENGOSUB5000
3900 GOTO16

4120 SYS(23162):IFR$="F"THENGOSUB5000
4130 GOTO3000
```

2 - Dopo aver visionato le videate di aiuto, al ritorno alla pagina di testo non ricompare la linea che indica il numero di colonna e riga. Per risolvere questo problema vanno battute queste linee di programma:

```
330  IFCH=30THEN1700
1995 GOTO16
```

Un ultimo chiarimento

Poiché il VIC 20 è configurabile con diverse quantità di RAM e ROM, alcuni lettori hanno avuto dei problemi con queste espansioni, erroneamente attribuiti alla stampante.
Ricordiamo che il programma Type Writer funziona esclusivamente nella seguente configurazione:

- VIC 20
- Espansione 16 Kbyte RAM (ne più ne meno!)
- Drive 1540, 1541 o registratore
- Stampante GP100VC (Seikosha), o MPS801 (Commodore)

Inoltre se si possiede un bus di espansione, non vi devono "assolutamente" essere inserite altre cartucce. Ad esempio la cartuccia Super Expander non è compatibile con il programma.
Per essere, comunque, certi che non fossero stati apportati cambiamenti al sistema operativo del VIC o delle stampanti, abbiamo testato il Type Writer con un nuovo VIC 20 e tutte le stampanti ad esso collegabili. Si è rilevato che il programma funziona perfettamente con la GP100, MPS801 e la vecchia VC 1515. Vi sono invece dei problemi con la nuova MPS802, con la quale è sconsigliato l'uso di questo programma.

Dimensionamento dinamico di matrici per C 64

Come noto il BASIC standard del C 64 prevede che lo statement Dim, usato per definire la dimensione di una matrice, possa essere eseguito una sola volta nell'ambito di un programma; se si tenta di ridimensionare una matrice, viene segnalato un Redim'd Array Error.
Cio può fornire talvolta un vantaggio dal punto di vista diagnostico, ma costituisce una forte limitazione in molti programmi che fanno uso di matrici o eseguono calcoli matriciali. L'impossibilità di ridimensionare le matrici obbliga il programmatore a prevedere all'inizio del programma il dimensionamento di ogni matrice per il massimo ordine possibile e ciò comporta in moltissimi casi un inutile spreco di memoria.
Il programma qui descritto consente di sostituire al comando Dim standard un nuovo comando Dim, che permette il dimensionamento dinamico di matrici; il nuovo Dim ridimensiona da capo la matrice indipendentemente dal fatto che essa sia o no stata già dimensionata in precedenza.
Se consideriamo, ad esempio, le seguenti linee:

```
10 DIM A(20,2),B(15)
..
..
50 DIM A(2,3,2),B(25,1,1,2)
```

Notiamo che la linea 10 dimensiona le matrici A e B. L'esecuzione della linea 50 col comando Dim standard causerebbe un Redim'd Array Error; col nuovo comando invece ridefinisce, da quel momento in poi, le matrici A e B secondo i nuovi indici racchiusi fra parentesi.

Type Writer e dimensionamento dinamico

Il principio di base del Dim dinamico

Nella memoria del C 64 le matrici vengono memorizzate in sequenza in un'area puntata dal vettore \$002F, secondo lo schema di figura 1.
Il contenuto del vettore \$002F è ADDR M_0 che rappresenta l'indirizzo dell'array M_0. I primi due byte contengono il nome della matrice M_0; i secondi due byte indicano la lunghezza l_0 di M_0, ovvero il numero di byte riservati per la memorizzazione di tutte le informazioni relative alla matrice M_0. In altre parole l'area di memoria (ADDR M_0)÷(ADDR $M_0 + l_0 - 1$) è riservata all'array M_0.
Dalla locazione ADDR M_1 = ADDR $M_0 + l_0$ sono memorizzate le informazioni relative alla matrice M_1 che occupa un'area di memoria di dimensione l_1 ... e così via.
Il vettore \$0031 punta alla fine dell'area dedicata alle matrici (ADDR FA).
Quando in un programma si incontra il comando Dim standard, l'interprete va a cercare se, fra i nomi delle matrici contenute nell'area matrici, c'è quello dell'array che si vuole dimensionare. Se non esiste apre una fine-

```
vettore $002F=ADDR M0
vettore $0031=ADDR FA
-----------------------------------------------
ADDR M0              x }
                     x } nome matrice
                     x }
                     x } l0= n.ro byte occupati da M0
                     •
                     •
                     •
-----------------------------------------------
ADDR M1              x }
(=ADDR M0 +l0 )      x } nome matrice
                     x }
                     x } l1 = n.ro byte occupati da M1
                     •
                     •
                     •
-----------------------------------------------
   • •
   • •
   • •
-----------------------------------------------
ADDR MN              x }
(=ADDR               x } nome matrice
  MN-1 +l N-1)       x }
                     x } lN= n.ro byte occupati da MN
                     •
                     •
                     •
-----------------------------------------------
ADDR FA
(=ADDR MN+ lN)
```

Figura 1. *Ecco come vengono memorizzate le matrici nel C 64.*

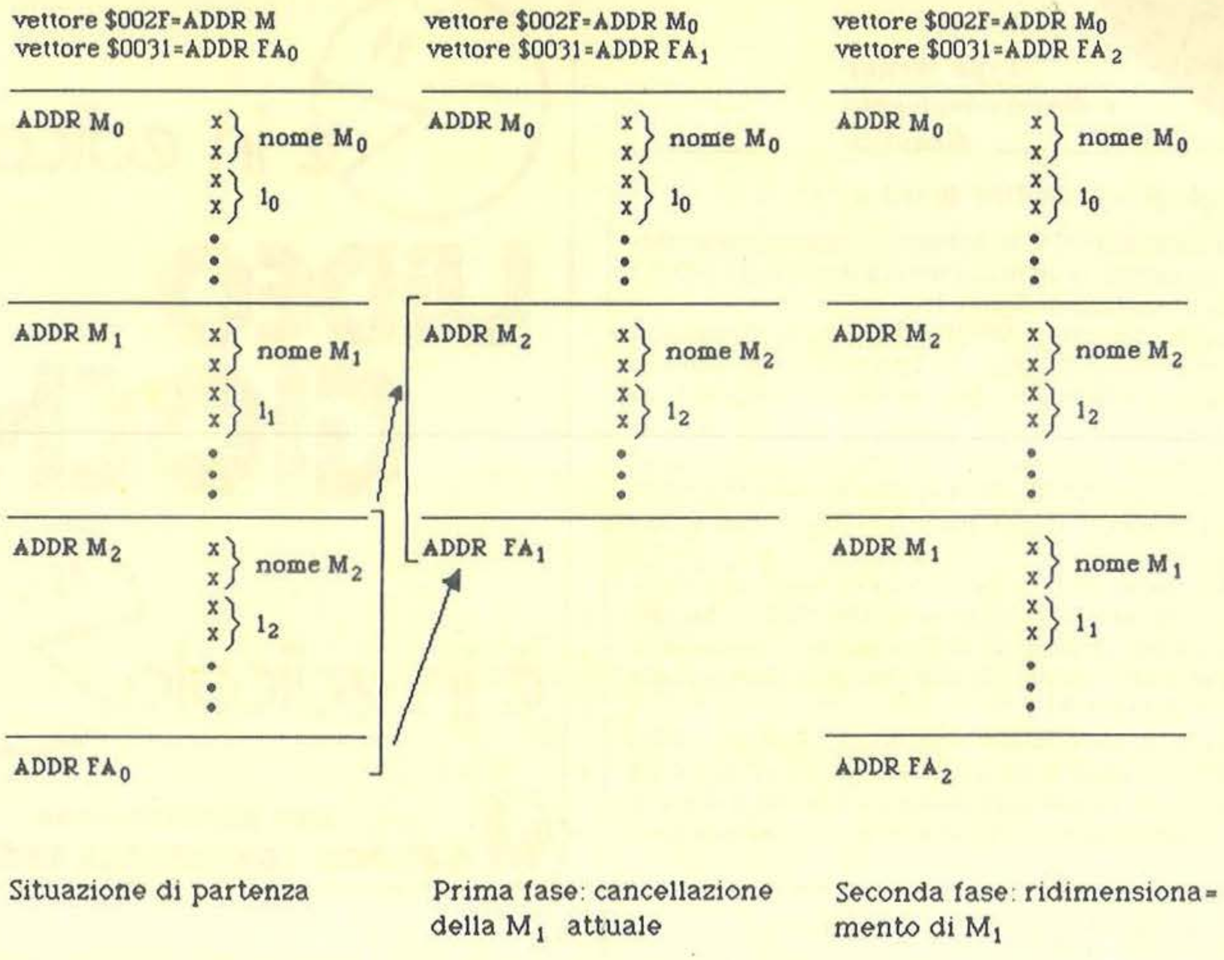


Figura 2. Schema di funzionamento del nuovo comando Dim.

stra al termine dell'area matrici, dedicata al nuovo array e aggiorna il contenuto del vettore $0031.
In caso contrario viene segnalato un Redim'd Array Error.
Per seguire il semplice principio su cui si basa la realizzazione del nuovo comando Dim si farà riferimento allo schema di figura 2.
Si supponga di voler ridimensionare la matrice M_1; il nuovo comando agisce in due tempi. Prima di tutto fa scomparire l'area dedicata a M_1 trasferendo tutto il blocco di memoria ADDR M_2 - ADDR FA_0 a partire da (ADDR M_0+l_0); a questo punto la matrice M_1 non esiste più fisicamente in memoria.
È dunque possibile eseguire un Dim standard che, non trovando il nome di M_1 fra quello delle matrici già in memoria, aprirà una finestra dedicata a M_1 al termine dell'area matrici e aggiornerà il contenuto del vettore $0031.

Il programma BASIC

Il listato 1 presenta il programma BASIC che sostituisce al vecchio il nuovo comando Dim.
Dopo il Run una scritta avverte di attendere circa 40 secondi per il caricamento e l'attivazione del nuovo comando.
La linea 20 ricopia l'interprete BASIC in RAM in modo da consentirne le necessarie modifiche (risiedendo l'interprete BASIC in ROM, non è possibile modificarlo direttamente).
La linea 30 modifica l'indirizzo del comando Dim nella tabella degli indirizzi.
Le linee 40, 42 eseguono le modifiche necessarie nella routine Dim standard.
La linea 50 scrive, a partire dall'indirizzo $C000, la nuova routine Dim.
Dopo aver effettuato un controllo sulla correttezza dei valori dei DATA letti nel programma (linea 52), la linea 60 seleziona l'area RAM in cui è scritto il nuovo interprete e, dopo averne comunicato l'abilitazione, libera completamente l'area BASIC che resta interamente a disposizione dell'utente (la nuova routine occupa 140 byte di memoria nell'area $C000 ÷ $C08C).
Nel listato 2 è riportato il disassemblato della nuova routine con i relativi commenti: bibliografia essenziale "Il S.O. del CBM 64" della EVM.
Si tenga presente che è possibile tornare in qualsiasi momento al BASIC originale digitando Poke 1,55 oppure premendo contemporaneamente i tasti Run/Stop e Restore.
Per ripassare dal BASIC originale a quello modificato, occorre invece digitare Poke 1,54.

Type Writer e dimensionamento dinamico

Listato 1 - *Il programma BASIC.*

```
0 REM ******************************
1 REM *                            *
2 REM *** DIM DINAMICO PER C.64 ***
3 REM *    -------------------     *
4 REM *            DI              *
5 REM * MAURIZIO PAOLINELLI        *
6 REM * VIA MAGELLANO, 15          *
7 REM * 20094 CORSICO  MI          *
8 REM * TEL. (02)4407707           *
9 REM *                            *
10 REM*****************************

12 PRINT"[<1CLR>][<1CRSR D>]"SPC(5):FORI=
0TO28:PRINT"*";:NEXT:PRINT
14 PRINTSPC(5)"*** DIM DINAMICO PER C.64
***"
16 PRINTSPC(5):FORI=0TO28:PRINT"*";:NEXT:
PRINT
18 PRINT"[<1CRSR D>]ATTENDI CIRCA 40 SECO
NDI, PREGO!"
19 REM *** RICOPIA BASIC IN RAM ***
20 FORI=40960TO49151:POKEI,PEEK(I):NEXT
29 REM *** MODIFICA INDIRIZZO DIM ***
30 POKE40984,2:POKE40985,192
39 REM *** MODIFICA DIM STANDARD ***
40 POKE45649,76:POKE45650,21:POKE45651,19
2
42 POKE45182,76:POKE45183,0:POKE45184,192
49 REM *** LETTURA NUOVA ROUTINE ***
50 A=0:FORI=49152TO49292:READN:A=A+N:POKE
I,N:NEXT
52 IFA<>17965THENPRINT"[<1CLR>][<1CRSR D>
]ERRORE NELLE ISTRUZIONI DATA!":END
59 REM *** ABILITA IL NUOVO BASIC ***
60 POKE1,54:PRINT"[<1CLR>][<1CRSR D>]NUOV
O BASIC ABILITATO!":NEW
70 END
99 *** NUOVA ROUTINE DIM ***
100 DATA32,253,174,165,122,141,19,192,165
,123,141,20,192,32,121,0,76
110 DATA129,176,234,234,208,17,32,148,177
,165,11,160,4,209,95,208,3
120 DATA76,234,178,76,69,178,200,177,95,1
33,88,24,101,95,133,90,200
130 DATA177,95,133,89,101,96,133,91,56,16
5,49,229,90,133,34,165,50
140 DATA229,91,133,35,170,160,0,240,14,17
7,90,145,95,200,208,249,230
150 DATA88,230,96,202,208,242,165,34,240,
9,177,90,145,95,200,196,34
160 DATA208,247,56,165,49,229,88,133,49,1
65,50,229,89,133,50,104,104
170 DATA104,104,198,11,208,250,173,19,192
,133,122,173,20,192,133,123,32
180 DATA121,0,76,129,176
```

Listato 2 - *Il disassemblato commentato della nuova routine.*

```
                    Comando "DIM" dinamico.

C000 JSR $AEFD      SYNTAX Error se il carattere non è ",".
C003 LDA $7A        Indirizzo carattere corrente
C005 STA $C013
C008 LDA $7B
C00A STA $C014      in $C013, $C014.
C00D JSR $0079      Prendi il carattere corrente
C010 JMP $B081      e salta al comando "DIM" standard.

C013 00  00         Registri di memoria indirizzo car. corrente.

                    Rientra dal comando "DIM" standard se
                    l'array da dimensionare era già stato
                    dimensionato in precedenza.
                    Contenuto dei registri:
                    A = $0C
                    Y = #$02
                    $5F, $60 = indirizzo inizio array da ridimensionare.

C015 BNE $C028      Se si tratta di dimensionamento, vai a ... $C028.
C017 JSR $B194      Calcola il puntatore al corpo dell'array.
C01A LDA $0B        Se il numero di dimensioni
C01C LDY #$04
C01E CMP ($5F),Y    è diverso da quello dichiarato
C020 BNE $C025      allora salta a ... $C025;
C022 JMP $B2EA      altrimenti calcola il riferimento all'elemento
C025 JMP $B245      BAD SUBSCRIT Error.                dell'array.

                    Procedura di ridimensionamento.

C028 INY            Memorizza in $58, $59 la lunghezza
C029 LDA ($5F),Y    della matrice da ridimensionare, e
C02B STA $58        in $5A, $5B l'indirizzo iniziale della
C02D CLC
C02E ADC $5F
C030 STA $5A
C032 INY
C033 LDA ($5F),Y
C035 STA $59
C037 ADC $60
C039 STA $5B        array successiva.
C03B SEC            Memorizza in $22, $23 la
C03C LDA $31
C03E SBC $5A
C040 STA $22
C042 LDA $32
C044 SBC $5B
C046 STA $23        lunghezza dell'area da tracferire.
C048 TAX            X = n.ro di pagine da trasferire.
C049 LDY #$00       Y = #$00.

                    Trasferimento del blocco di memoria.
                    $5A, $5B = indirizzo inizio sorgente.
                    $31, $32 = indirizzo fine sorgente.
                    $5F, $60 = indirizzo inizio dèstinazione.
                    X = n.ro pagine (256 bytes) da trasferire.
                    $22 = resto da trasferire.
                    Il byte più basso viene trasferito per primo.

C04B BEQ $C05B      Se lunghezza area <256, trasferisci il resto.
C04D LDA ($5A),Y    Prendi il carattere dalla sorgente
C04F STA ($5F),Y    e memorizzalo a destinazione.
C051 INY
C052 BNE $C04D      Continua fino a fine pagina.
C054 INC $58        Aggiorna indirizzo pagina sorgente
C056 INC $60        e indirizzo pagina destinazione.
C058 DEX
C059 BNE $C04D      Esegui per tutte le pagine da trasferire.
C05B LDA $22        Se il resto è = 0,
C05D BEQ $C068      fine trasferimento.
C05F LDA ($5A),Y    Prendi il carattere dalla sorgente
C061 STA ($5F),Y    e memorizzalo a destinazione.
C063 INY
C064 CPY $22        Esegui fino alla
C066 BNE $C05F      fine del resto.

C068 SEC            Aggiorna il vettore $0031
C069 LDA $31        che punta all'indirizzo
C06B SBC $58
C06D STA $31
C06F LDA $32
C071 SBC $59
C073 STA $32        di fine array.

C075 PLA            Svuota lo stack
C076 PLA            a seconda del
C077 PLA
C078 PLA
C079 DEC $0B
C07B BNE $C077      n.ro di dimensioni dell'array.
C07D LDA $C013      Ripristina in $7A, $7B
C080 STA $7A        l'indirizzo del
C082 LDA $C014
C085 STA $7B        carattere corrente del "buffer".
C087 JSR $0079      Prendi il carattere corrente
C08A JMP $B081      ed esegui il "DIM" standard
                    (ridimensiona l'array).
```

I SEGRETI DEI PERSONAL

TEXAS TI99/4A

Recupero degli errori

di Sergio Borsani

Un buon programma dovrebbe essere in grado di sopravvivere a tutti i tentativi, più o meno consapevoli, di portarlo in una condizione d'errore. In un certo senso il programmatore deve prevedere ogni possibile risposta, da parte dell'utente, che possa causare la comparsa di un messaggio d'errore e il conseguente blocco del programma. In questo compito, non sempre facile, il TI Extended BASIC viene in aiuto con due istruzioni: On Error e Call Err.

La prima esiste in due formati: On Error Stop e On Error n, dove n è un appropriato numero di linea. La parola chiave Stop ha valore di default e fa in modo che appaia il solito messaggio dopo il quale il programma si arresta; nè più nè meno di quello che avviene normalmente. Ben diversamente agisce il secondo formato. Esso permette di inviare il controllo ad una subroutine, come fa una Gosub, in modo da poter visualizzare un qualsiasi messaggio e poi procedere nell'esecuzione del programma.

L'istruzione On Error n deve essere ripetuta ogni volta che si incappa in un errore per cui il punto più ovvio dove inserirla in un programma è il menu principale che costituisce un passaggio obbligato. La routine che gestisce l'errore, come tutte le subroutine, termina con l'istruzione Return. Tuttavia, a differenza di quanto avviene con una Gosub, lo statement Return può assumere tre formati: Return da solo, Return + numero di linea e Return Next. Return fa tornare il programma alla linea che ha causato l'errore. Return + numero di linea trasferisce il controllo a quel numero di linea. Return Next fa riprendere la esecuzione del programma dalla linea successiva a quella che ha causato l'errore. Nel caso si specifichi un numero di linea, il programma potrebbe opportunamente tornare al menu dove verrebbe eseguita nuovamente l'istruzione On Error prima di proseguire in una qualsiasi sezione di lavoro.

Naturalmente questa è solo una proposta e nulla vieta una diversa strutturazione.

La subroutine che viene chiamata in caso di errore, invece di limitarsi ad inviare un messaggio generico, può essere potenziata e resa in grado di specificare all'utente il tipo di errore e la linea di programma nella quale si è verificato. Ciò è reso possibile dall'uso dell'istruzione Call Err. La sintassi completa è: Call Err (codice, tipo [,x,numero di linea]). Quando nel programma viene commesso un errore e viene eseguita una Call Err la prima variabile numerica tra parentesi contiene il codice dell'errore. Il suo significato è reso esplicito nell'Appendice N del manuale del BASIC esteso, a partire da pag. 212. Se si caricano in una tabella alcuni codici ed i corrispondenti messaggi d'errore, la subroutine diventa in grado di indicare all'utente il tipo di errore che è stato commesso. La seconda variabile tra parentesi assume solitamente un valore negativo. Se il codice d'errore è il 130, allora essa indica il numero del file causa dell'errore. La terza variabile ha sempre il valore 9.

La quarta, infine, specifica il numero della linea di programma nella quale si è verificato l'errore. Le ultime due variabili tra parentesi sono facoltative, pertanto

Listato 1. *Il programma Recupero degli errori.*

```
100 REM ******************
110 REM *                *
120 REM *    RECUPERO    *
130 REM *  DEGLI ERRORI  *
140 REM *                *
150 REM ******************
160 DATA 109,FILE ERROR
170 DATA 24,STRING-NUMBER MISMATCH
180 DATA 14,SYNTAX ERROR
190 DATA 60,LINE NOT FOUND
200 DATA 130,I/O ERROR
210 FOR J=1 TO 5 :: READ EC(J),EM$(J)::
NEXT J
220 CALL CLEAR :: ON ERROR 540
230 DISPLAY AT(2,1):"IL PROGRAMMA E' DIV
ISO IN 6 SEZIONI OGNUNA CONTENENTE UNDIV
ERSO TIPO DI ERRORE."
240 DISPLAY AT(8,1):"SCEGLI LA SEZIONE."
250 FOR RIGA=10 TO 20 STEP 2 :: N=RIGA/2
-4 :: DISPLAY AT(RIGA,1):STR$(N);". SEZI
ONE ";CHR$(64+N):: NEXT RIGA
260 DISPLAY AT(24,10):"SCELTA?" :: ACCEP
T AT(24,18)SIZE(1)VALIDATE("123456"):SC$
270 ON VAL(SC$)GOTO 280,310,340,370,400,
430
280 CALL CLEAR :: DISPLAY AT(1,1):"QUEST
A SEZIONE CONTIENE UN  ERRORE DI INPUT/O
UTPUT." :: GOSUB 530
290 OPEN #4:"IEEE 488" :: PRINT #4:"TI-9
9/4A"
300 END
310 CALL CLEAR :: DISPLAY AT(1,1):"QUEST
A SEZIONE CONTIENE UNA ISTRUZIONE DOVE V
IENE CONFU-SA UNA STRINGA CON UN NUMERO"
 :: GOSUB 530
320 A$=99999
330 END
340 CALL CLEAR :: DISPLAY AT(1,1):"IN QU
ESTA SEZIONE SI TENTA  DI LEGGERE DATI D
A UN FILE  CHE NON ESISTE." :: GOSUB 530
350 INPUT #1:A$
360 END
370 CALL CLEAR :: DISPLAY AT(1,1):"IN QU
ESTA SEZIONE UNA GOSUB MANDA IL PROGRAMM
```

```
A AD UNA   LINEA INESISTENTE." :: GOSUB
530
380 GOTO 32767
390 END
400 CALL CLEAR :: DISPLAY AT(1,1):"IN QU
ESTA SEZIONE VIENE USA-TA COME VARIABILE
 UNA PAROLARISERVATA." :: GOSUB 530
410 MAX=999
420 END
430 CALL CLEAR :: DISPLAY AT(1,1):"IN QU
ESTA SEZIONE SI PUO'   VERIFICARE UNA CO
NDIZIONE DIERRORE CON LA FUNZIONE LOG."
440 DISPLAY AT(6,1):"ESSA SI VERIFICA QU
ANDO SI  PONE A<0."
450 DISPLAY AT(10,1):"SCRIVI UN NUMERO."
460 DISPLAY AT(12,1)BEEP:"A =" :: ACCEPT
 AT(12,5)VALIDATE(NUMERIC):A
470 L=LOG(A)
480 DISPLAY AT(14,1):"IL LOGARITMO DI";A
;"E'";L
490 DISPLAY AT(24,2):"- VUOI CONTINUARE?
 (Y/N) -"
500 CALL KEY(0,K,S):: IF S=0 THEN 500
510 IF K=89 OR K=121 THEN 430
520 IF K=78 OR K=110 THEN 220 ELSE 500
530 FOR TIME=1 TO 1000 :: NEXT TIME :: R
ETURN
540 CALL ERR(X,Y,W,Z):: FOR J=1 TO 5 ::
IF X=EC(J)THEN 560
550 NEXT J :: DISPLAY AT(20,1):"* ERRORE
!" :: GOTO 570
560 DISPLAY AT(20,1):"* ";EM$(J)
570 DISPLAY AT(21,3):"ALLA LINEA";Z
580 DISPLAY AT(24,6):"<PREMI UN TASTO>"
590 CALL KEY(0,K,S):: IF S=0 THEN 590 EL
SE RETURN 220
```

l'istruzione può essere scritta o nella forma Call Err (X, Y) oppure nell'altra Call Err (A,B,C,D).
Non è superfluo ricordare che le istruzioni On Error e Call Err formano un abbinamento veramente utile al programmatore nella fase di debug di un programma. Infatti, se si dovesse subire uno stop ad ogni errore, si dovrebbe ripartire con il Run azzerando così tutte le variabili presenti in quel momento con il risultato che si dovrebbero introdurre nuovamente i dati di prova. Recuperando gli errori si può invece proseguire con i controlli e le verifiche senza perdere tempo prezioso.
Il breve programma del listato 1 vuole semplicemente schematizzare i concetti su esposti mostrando come per alcuni errori possa essere mandato un messaggio specifico, per altri un messaggio generico. Il programma è strutturato in sei parti selezionabili dal menu; ognuna contiene un diverso errore.
Nell'ultima, la sesta, l'errore è conseguente al valore inserito dall'utente in input; infatti la funzione matematica Log, lì presente, non ammette argomenti negativi o uguali a zero.

Le istruzioni Data poste nella parte iniziale del programma contengono alcuni codici ed i corrispondenti messaggi d'errore. Alla linea 210 i dati vengono caricati in due vettori.
Le linee 220-270 costituiscono il menu e contengono l'istruzione On Error 540. Seguono le sei sezioni con altrettanti errori che altrimenti avrebbero causato uno stop del programma. La subroutine che li gestisce inizia alla linea 540 ed occupa tutta la parte restante, fino al termine. La stessa linea 540 contiene l'istruzione Call Err ed il ciclo per la ricerca in tabella del codice individuato dalla variabile X.

SINCLAIR ZX SPECTRUM

Extended BASIC

di Marcello Spero

Fra le possibilità offerte dall'Interfaccia 1, il cui scopo principale è quello di controller per i microdrive, c'è anche quella, meno nota, di estensione del vocabolario BASIC dello Spectrum.
Quello che vedremo questa volta è un metodo per ottenere il medesimo risultato senza l'ausilio dell'interfaccia.
È appena il caso di ricordare i vantaggi offerti dalla possibilità di disporre di un certo numero di istruzioni su misura, create apposta per risolvere i nostri problemi specifici.
Istruzioni per la grafica, tool kit, aiuti per la verifica e la correzione dei programmi, non sono che un pallido esempio.
Per ragioni che vedremo in seguito, tutti i nostri nuovi comandi ed istruzioni devono avere la caratteristica di provocare immediatamente (cioè con il loro primo carattere) un errore al normale controllo di sintassi operato dall'interprete in ROM. Due sono i metodi possibili per ottenere questo.
Uno, più semplice, consiste nello scegliere come primo carattere una delle parole BASIC, da Rnd a Step, ottenibili per mezzo del Symbol Shift. In questo modo si aggira il modo "K", e si ottiene un sicuro errore.
Il secondo metodo, più elegante ma più complesso, consiste nell'iniziare la nostra istruzione con un carattere che provochi l'uscita dal modo "K", e quindi comporre il nome della nuova istruzione lettera per lettera.

Un'istruzione che provoca lo "scrolling" laterale dello schermo, ad esempio, può essere realizzata con:

STEP d,5

(per uno spostamento verso destra di 5 pixel, ad esempio) con il primo metodo, o con:

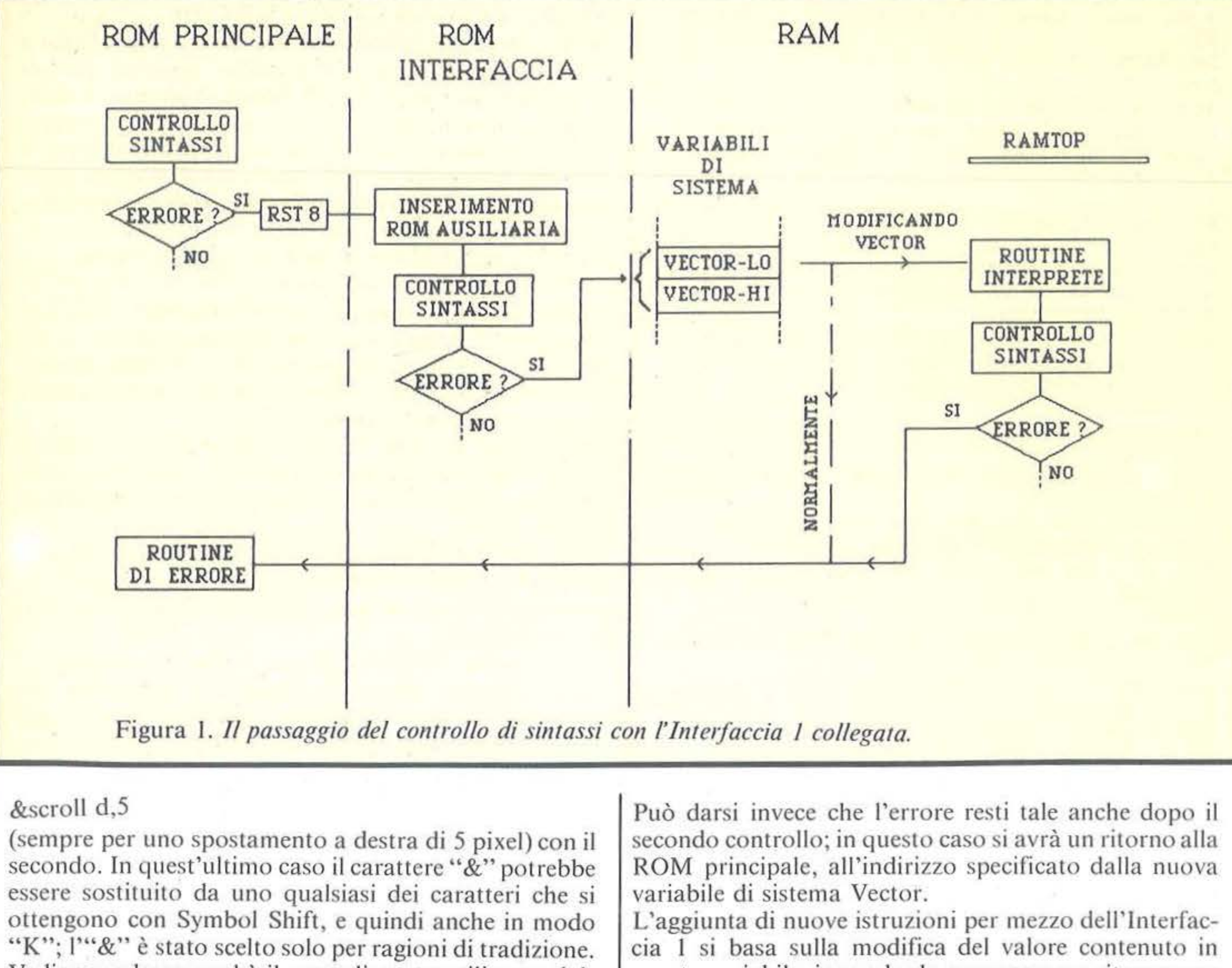


Figura 1. *Il passaggio del controllo di sintassi con l'Interfaccia 1 collegata.*

&scroll d,5
(sempre per uno spostamento a destra di 5 pixel) con il secondo. In quest'ultimo caso il carattere "&" potrebbe essere sostituito da uno qualsiasi dei caratteri che si ottengono con Symbol Shift, e quindi anche in modo "K"; l'"&" è stato scelto solo per ragioni di tradizione.
Vediamo adesso qual è il procedimento utilizzato dall'interfaccia 1 per consentire all'utilizzatore di inserire le proprie estensioni al BASIC, poiché è proprio emulando questo procedimento che si giunge ad ottenere la stessa possibilità sulle macchine sprovviste di tale interfaccia.
La caratteristica comune a tutte le nuove istruzioni o comandi deve essere abbiamo detto, quella di provocare un errore durante l'esame da parte dell'interprete BASIC. Il verificarsi di un errore, infatti, provoca sempre l'esecuzione di una Rst 8, istruzione in linguaggio macchina che provoca l'inserimento dell'Interfaccia 1. A questo punto l'istruzione risultata errata viene nuovamente esaminata, questa volta dall'interprete contenuto nella ROM dell'interfaccia. Se a questo secondo esame risulterà corretta (come avverrà nel caso delle nuove istruzioni da usare con l'interfaccia, come Cat, Format, Move, ecc.), l'istruzione sarà accettata, o nel caso si tratti di un comando diretto, eseguita.
Può darsi invece che l'errore resti tale anche dopo il secondo controllo; in questo caso si avrà un ritorno alla ROM principale, all'indirizzo specificato dalla nuova variabile di sistema Vector.
L'aggiunta di nuove istruzioni per mezzo dell'Interfaccia 1 si basa sulla modifica del valore contenuto in questa variabile, in modo da provocare un ritorno non alla ROM principale, in corrispondenza della routine di errore, ma ad un determinato indirizzo in RAM in cui avremo collocato un nostro interprete, che eseguirà un terzo esame dell'istruzione o comando incriminato.
Essendo la variabile Vector l'unico punto in cui l'interprete BASIC dello Spectrum è vettorizzato (possiede cioè per una sua routine un indirizzo modificabile), ed esistendo questa variabile solo con l'inserimento dell'Interfaccia 1, da molti è ritenuta impossibile l'estensione del BASIC senza l'aggiunta di hardware supplementare (l'Interfaccia 1, appunto).
Tale estensione è invece perfettamente possibile anche per coloro che non possiedono la famigerata (e costosa) interfaccia.
Per comprendere come ciò sia possibile, occorre esaminare brevemente la meccanica del controllo di sintassi operato dall'interprete BASIC, ed in particolare ciò che avviene in caso di errore, sempre nelle macchine sprov-

viste di Interfaccia 1 od al ritorno alla ROM principale, tramite Vector, dove l'interfaccia è presente.
Il verificarsi di un errore, sia durante il controllo di sintassi che in fase di esecuzione del comando o istruzione, provoca una serie di operazioni.
• L'indirizzo raggiunto dall'interprete, contenuto nella variabile di sistema CH ADD, è copiato nel puntatore di errore, la variabile X PTR.
• Il codice dell'errore viene posto nella variabile ERR NR. Il codice è sempre inferiore di uno a quello del messaggio corrispondente; sarà cioè 255 per 0, OK e 11 per C, Nonsense in BASIC. In particolare, qualsiasi comando o istruzione non riconosciuto produrrà un codice di errore uguale ad 11, e questo sia in fase di controllo della sintassi che in fase di esecuzione.
• Il puntatore alla catasta di sistema (cioè il registro SP dello Z80, lo stack pointer) viene caricato con il valore presente nella variabile di sistema ERR SP, venendo in tal modo a puntare al primo elemento della catasta.
• La catasta del calcolatore e l'area di lavoro del sistema vengono ripulite.
• L'istruzione in linguaggio macchina Ret rimuove dalla

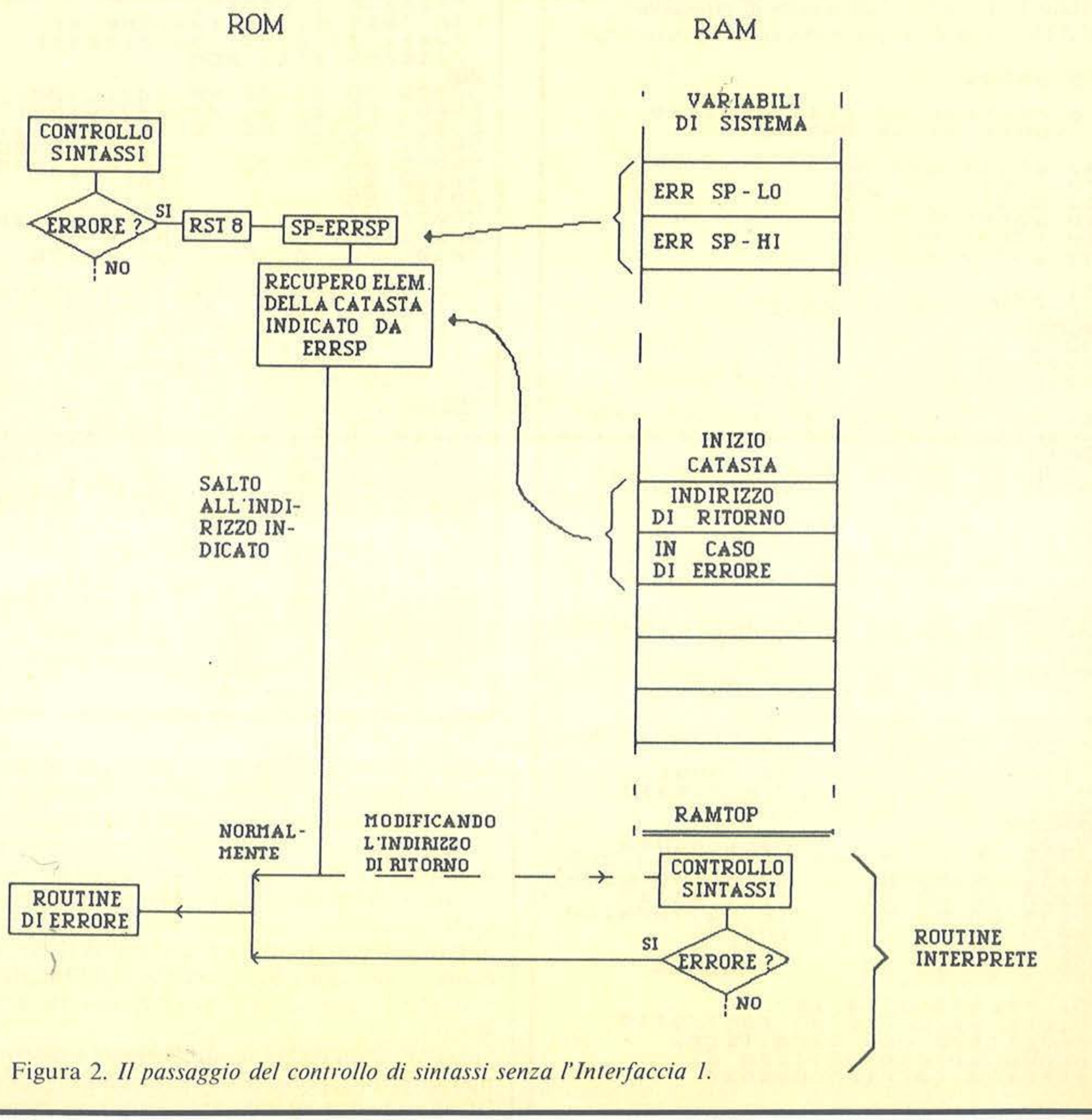


Figura 2. *Il passaggio del controllo di sintassi senza l'Interfaccia 1.*

catasta del sistema l'elemento cui punta SP, e compie un salto all'indirizzo da lui indicato.

Durante il controllo della sintassi, questo indirizzo è normalmente 12B7 Hex, che corrisponde alla serie di routine che provocano la visualizzazione della linea incriminata con un punto interrogativo lampeggiante in posizione appropriata.

Durante l'esecuzione, invece, questo indirizzo è 1303 Hex, corrispondente alla routine che produce il messaggio di errore.

Listato 1. *Assembly della routine di estensione del BASIC e dell'interprete per la nuova istruzione.*

```
org 50001

'La routine va posta subito al
di sopra della RAMTOP

equ 23610 ERR NR
equ 23645 CH ADD
equ 23641 E LINE
equ 23693 ATTR P
equ 23624 BORDCR
equ 23730 RAMTOP

'Il codice di errore e' 11
 (C NONSENSE IN BASIC)?
START
50001 3A 3A 5C      ld a,(ERR NR)
50004 FE 0B         cp 11
50006 28 23         jr z,ERRC

'Il bit 7 della variabile FLAGS
e' settato in fase di esecuzione
ERROR
50008 FD CB 01 7E bit 7,(iy+1)
50012 20 08         jr nz,RUNERR

'Errore di sintassi.
 L'indirizzo di inizio della
 routine (RAMTOP+1) e' posto
 sulla catasta, e si torna alla
 ROM
SINTERR
50014 2A B2 5C      ld hl,(RAMTOP)
50017 23            inc hl
50018 E5            push hl
50019 C3 B7 12      jp &12b7

'Errore in esecuzione.
 Viene prodotto il messaggio e
 sono rimosse le forme
 "floating point" dei numeri
 prima di eseguire le stesse
 operazioni di SINTERR
RUNERR
50022 CD 03 13      call &1303
50025 FD 36 00 FF ld (iy+0),255
50029 2A 59 5C      ld hl,(E LINE)
50032 CD A7 11      call &11a7
50035 2A B2 5C      ld hl,(RAMTOP)
50038 23            inc hl
50039 E5            push hl
50040 C3 B4 12      jp &12b4

'Si tratta di &cls?
 Viene eseguito un controllo
 carattere per carattere,
 dando errore in caso di
 mancata corrispondenza
ERRC
50043 2A 5D 5C      ld hl,(CH ADD)
50046 2B            dec hl
50047 22 5D 5C      ld(CH ADD),hl
50050 DF            rst 24
50051 FE 26         cp 38
50053 20 D1         jr nz,ERROR
50055 E7            rst 32
50056 FE 63         cp 99
50058 20 CC         jr nz,ERROR
50060 E7            rst 32
50061 FE 6C         cp 108
50063 20 C7         jr nz,ERROR
50065 E7            rst 32
50066 FE 73         cp 115
50068 20 C2         jr nz,ERROR
50070 E7            rst 32
50071 FE 0D         cp 13
50073 28 04         jr z,OK
50075 FE 3A         cp 58
50077 20 B9         jr nz,ERROR

'Se la parola e' &cls ed e'
 seguita da ENTER o ":" le
 variabili ERR NR e X PTR
 sono aggiornate ad indicare
 "nessun errore".
 In fase di esecuzione si
 passa ad EXEC, altrimenti
 ritorno alla ROM.
OK
50079 FD 36 00 FF ld(iy+00),255
50083 FD 36 26 00 ld(iy+38),0
50087 FD CB 01 7E bit 7,(iy+1)
50091 20 0C         jr nz,EXEC
50093 2A B2 5C      ld hl,(RAMTOP)
50096 23            inc hl
50097 E5            push hl
50098 21 B7 12      ld hl,&12b7
50101 E5            push hl
50102 C3 76 1B      jp &1b76

'Le variabili che contengono
 gli attributi di schermo e
 border vengono modificate
 opportunamente, e quindi
 sono eseguiti i comandi
 BORDER e CLS.
EXEC
50105 3E 07         ld a,7
50107 32 8D 5C      ld (ATTR P),a
50110 32 48 5C      ld (BORDCR),a
50113 3E 00         ld a,0
50115 D3 FE         out (254),a
50117 CD 6B 0D      call &0d6b

'Ritorno alla ROM dopo aver
 ripristinato in catasta
 l'indirizzo della routine
50120 2A B2 5C      ld hl,(RAMTOP)
50123 23            inc hl
50124 E5            push hl
50125 C3 76 1B      jp &1b76
```

Appare chiaro a questo punto che la presenza all'interno della catasta di sistema dell'indirizzo di errore rappresenta un analogo della variabile Vector, e quindi di fatto una vettorizzazione dell'interprete principale.

Sostituendo questo indirizzo con quello di una nostra routine in linguaggio macchina con funzione di interprete per quanto riguarda la o le nuove istruzioni da noi introdotte, introdurremo la possibilità di un secondo esame, esattamente come avviene con l'Interfaccia 1.

Il listato 1 chiarirà senz'altro meglio tutto il procedimento.

Si tratta infatti della routine destinata a determinare il tipo di errore, provocando altrimenti un ritorno alla ROM principale all'indirizzo opportuno, diverso a se-

conda che il sistema si trovi in fase di controllo di sintassi o in fase di esecuzione. Prima di provocare il ritorno vero e proprio, inoltre, la routine colloca nuovamente sulla catasta di sistema il proprio indirizzo, in modo da assicurare il proprio inserimento al successivo errore.
La parte relativa al controllo di sintassi ed all'esecuzione della o delle nuove istruzioni è invece variabile, a seconda di ciò che si vuole ottenere. Come esempio è stata inserita una routine che interpreta ed esegue un'istruzione molto semplice:

&cls

Questa nuova istruzione è stata costruita a somiglianza di:

CLS #

presente nel set esteso del BASIC dell'Interfaccia 1, che oltre a ripulire lo schermo ne ripristina le condizioni definite "normali": sfondo bianco, caratteri neri, lampeggio e luminosità a zero.

La nostra nuova istruzione si comporta in modo simile, tranne per il fatto che le condizioni di schermo introdotte sono: sfondo nero, caratteri bianchi. Luminosità e lampeggio vengono sempre azzerati.
La scelta di questa combinazione di schermo non è casuale, ma riflette la constatazione che caratteri chiari su di uno sfondo scuro danno una visione molto riposante, per lo meno utilizzando un televisore.
Come potete notare esaminando il listato 1, la parte più complessa è quella relativa alla verifica della sintassi, dovendo questa essere fatta lettera per lettera. Per ottenere, uno dopo l'altro, i caratteri che compongono l'istruzione, è stata usata la medesima routine utilizzata dall'interprete "vero" per questa operazione.
Questa routine possiede due punti di ingresso, accessibili con le istruzioni Rst 24 ed Rst 32 (che equivalgono a Rst 18 ed Rst 20 espresse in esadecimale).
Con il primo si ottiene di ritorno, nel registro A dello Z80, il codice del carattere indirizzato dalla variabile di sistema CH ADD o, se questo non è utile ai fini dell'interpretazione (spazio, codici di colori, ecc.), del primo carattere utile. In questo caso CH ADD viene fatta avanzare.
Con il secondo punto di ingresso si ottiene invece un avanzamento preliminare di CH ADD, prima di eseguire le medesime operazioni di recupero ed eventuale avanzamento.
Poichè, al momento del verificarsi dell'errore che trasferisce il controllo alla nostra routine, CH ADD punta sempre al carattere successivo a quello che ha prodotto l'errore, basterà decrementare CH ADD di uno per ottenere la posizione del primo carattere da controllare, che verrà recuperato con Rst 24. I successivi caratteri saranno invece recuperati da altrettante Rst 32.

Per ciascun carattere recuperato viene eseguito un controllo, fino ad arrivare al carattere Enter o ":" che deve chiudere l'istruzione. In caso di mancata corrispondenza viene fatto ritorno alla routine di errore.
Al termine, se tutto è andato per il verso giusto, le variabili di sistema ERR NR e X PTR, indirizzate rispettivamente da (IY+0) e (IY+38), vengono aggiornate ad indicare la condizione di "non errore". Quindi, se la variabile Flags, indirizzata da (IY+l) indica con il suo bit 7 la condizione di esecuzione, viene eseguito un salto alla routine relativa. In caso contrario il controllo torna alla ROM, che provvede ad inserire la nuova linea nel programma.
La parte dedicata all'esecuzione è davvero semplicissima. Si riduce infatti alla modifica del valore delle variabili ATTR P e BORDCR, contenenti gli attributi per schermo e border, ed all'esecuzione del cambio di border seguita da un Cls.
Al termine il controllo torna alla ROM, dopo aver preparato la catasta per un successivo errore.
Il listato 2 contiene il codice macchina, di tipo autorilo-

Listato 2. *Codice macchina della routine.*

```
 58, 58, 92,254, 11, 40, 35,253,
203,  1,126, 32,  8, 42,178, 92,
 35,229,195,183, 18,205,  3, 19,
253, 54,  0,255, 42, 89, 92,205,
167, 17, 42,178, 92, 35,229,195,
180, 18, 42, 93, 92, 43, 34, 93,
 92,223,254, 38, 32,209,231,254,
 99, 32,204,231,254,108, 32,199,
231,254,115, 32,194,231,254, 13,
 40,  4,254, 58, 32,165,253, 54,
  0,255,253, 54, 38,  0,253,203,
  1,126, 32, 12, 42,178, 92, 35,
229, 33,183, 18,229,195,118, 27,
 62,  7, 50,141, 92, 50, 72, 92,
 62,  0,211,254,205,107, 13, 42,
178, 92, 35,229,195,118, 27

CHECKSUM:  14546
```

cante, a condizione che lo si collochi subito al di sopra della RAMtop. Si tratta di 127 byte che possono essere agevolmente caricati per mezzo del programmino che vedete nel listato 3. Questo programma esegue anche il controllo sul checksum (cioè la somma dei codici introdotti), in modo da evidenziare eventuali errori.
Il listato 4 contiene invece il programma di predisposizione della catasta. Va fatto eseguire dopo aver caricato il codice macchina per rendere possibile la chiamata della routine, che in seguito provvederà da sola al mantenimento di questa condizione.

Dopo aver caricato il codice, fate girare il programma e quindi provate la nuova istruzione.
$cls

Listato 3. *Programma di caricamento del codice macchina.*

```
  1 REM ************************
        *                      *
        *      PROGRAMMA       *
        *          DI          *
        *     CARICAMENTO      *
        *   CODICE MACCHINA    *
        *                      *
        ************************

 10 LET rtp=PEEK 23730+256*PEEK
    23731
 20 LET check=0
 30 FOR i=rtp+1 TO rtp+127
 40 INPUT k
 50 POKE i,k
 60 PRINT i,PEEK i
 70 LET check=check+PEEK i
 80 NEXT i
 90 INPUT "CHECKSUM?",csum
100 IF csum=check THEN STOP
110 PRINT TAB 13; FLASH 1;
    "ERRORE"'' FLASH 0;TAB 4;
    "RIPETERE IL CARICAMENTO"
120 PRINT #0;TAB 7;
    "premere un tasto"
130 PAUSE 0
140 CLS : GO TO 20
```

Listato 4. *Programma di predisposizione della catasta.*

```
  1 REM ************************
        *                      *
        *      PROGRAMMA       *
        * DI  PREDISPOSIZIONE  *
        *                      *
        ************************

 10 LET start=PEEK 23730+256*
    PEEK 23731+1
 20 LET errsp=o23613+256*
    PEEK .23614
 30 POKE errsp,
    start-256*INT (start/256)
 40 POKE errsp+1,
    INT (start/256)
```

che dovrebbe produrre uno schermo nero e pulito, nonché un border nero.

Volendo a questo punto eliminare il programma di predisposizione, cancellare linea per linea senza dare New. Ci sono infatti delle istruzioni che provocano il ripristino delle condizioni originali di catasta; sono:

New, Clear, Run

Ogni volta che le userete dovrete procedere nuovamente alla modifica dall'esterno della catasta. ■

PICCOLI ANNUNCI

Apple

Vendo per Apple II **"Supertoto 1.0"**, superprogramma Totocalcio inedito. Tre diverse opzioni di selezioni incrociate (nr. segni 1X2 consecutivi, correzione errori) con output nr. colonne utili a L. 60.000 con manuale. Roberto Rossi - Via Lario, 26 - 20159 Milano - Tel. 02/6070236 (ore serale)

Cambio **software** per Apple II, assicuro una risposta a chiunque mi invia la sua lista. Guido Pascale - Via Pasteur, 24 - 34139 Trieste

Compro **software vario** per Apple IIc. Inviare elenchi a: Angelo Sala - Via Olmi, 7 - 20077 Melegnano (MI)

Cerco qualsiasi programma per Apple **che tratti l'argomento Baseball - Calcio - Football americano.** Cerco anche il programma Crush Crumble and Chomp. Giancarlo Fedel - Via Marconi, 29 bis - 34018 Staranzano (GO) - Tel. 710127

Occasione. Vendo software per Apple II. Dispongo di **più di 80 programmi** tra giochi e utilities (The Last One - Apple Writer II - Flight Simulator - ecc.). Disponibili su disco, alcuni con listato. Invio gratuitamente la lista dei prezzi. Luigi Noschese - Via Bellini, 12 - 84098 Pontecagnano (SA) - Tel. 098/381121

Per Apple II cambio, vendo **programmi di ogni genere.** Concordabili pacchetti di programmi a prezzi speciali. Vendo HP41C più modulo RAM (totale 127 reg o 889 byte) a L. 300.000. Maurizio Mellone - Via Sabbionara, 9 - 36061 Bassano Del Grappa (VI) - Tel. 0424/20015

Vendo **programmi di ogni genere** per Apple II a L. 10.000 cadauno, compreso il dischetto. Richiedere la lista completa di oltre 1.000 titoli. Software per progettisti edili a prezzi contenuti. Raffaele Castelli - Via dei Campari, 3 - 24020 Gorno (BG)

■ Valorizzate il vostro Apple con **programmi altamente professionali che soddisfino ogni vostra esigenza.** Massima serietà, garanzia e celerità nel fornire il materiale richiesto. Luigi Palumbo - Via A. Ristori, 8 - 00197 Roma - Tel. 06/802783 ■

Per Apple II cerco **i seguenti programmi:** Logica Simbolica, Legionaire, War In Russia, Knight Of The Desert, V.C., Diet Analysis, Tridi, Programmare in Pascal, Anatomy 1, educazionali di fisica e programmi sull'Intelligenza Artificiale. Antonio Scala - Via G. Imbroda, 39 - 80035 Nola (NA) - Tel. 081/8234710

Cambio software per Apple II - IIe - IIc. Dispongo di **oltre 400 programmi;** mandatemi la vostra lista e vi invierò la mia. Rispondo a tutti. Isabella Bottini - Via Galilei, 681 - 18038 Sanremo (IM) - Tel. 0184/882095

Cambio, vendo **oltre 400 programmi** per Apple. Possiedo data base, molti giochi, gestionali. Scrivere a: Paolo Grandicelli - Casella Postale 66 - 62012 Civitanova M. (MC) - Tel. 0733/74369

A.A.A. Affaroni - Vendo **200 programmi Apple professionali** a L. 500.000. Carlo Cocciazucca - Via Montesecco, 15 - 65010 Spoltore (PE) - Tel. 085/207466

Cambio **software** per Apple. Cerco manuale del programma "The Graphic Solution". Annuncio valido per le province di Terni, Perugia, Macerata, Ascoli Piceno e Ancona. Giuseppe Albucci - Via Marzabotto, 16 - 05100 Terni - Tel. 0744/814147

Vendo per Apple II **Koala Pad usata solo due settimane** con manuali e dischetto originali a L. 300.000. Per informazioni rivolgersi a: Luigi Tolomelli - Via Martini, 15 - 51016 Montecatini Terme (PT) - Tel. 0572/73175

■ Vendo **scheda 80 colonne** per Apple IIe al miglior offerente. Tel. 02/498226 ■

Commodore

Cambio, vendo **circa 1.000 programmi** per C64, fra i quali Decathlon, Pitfall, Beamrider, River Raid e altri magnifici giochi; utility, gestionali; uno più bello dell'altro. Solo Milano e zone limitrofe. Angelo Settembrini - Via Cassanese, 194 - 20090 Segrate (MI) - Tel. 02/2136514 (dopo le 17.30)

Compro, cambio, vendo **programmi** per C64, cerco giochi ed utilities. Inviate la vostra lista con i relativi prezzi. Ringrazio in anticipo coloro che mi scriverenno. Giovanni Marino - Via Ragnaia, 28 - 57012 Castiglioncello (LI) - Tel. 0586/753017

Vendo **programmi** per Commodore 64 a L. 3.000 cadauno. Richiedere lista telefonando o scrivendo a: Borghesi Simone - Via Monte Santo, 17 - 53036 Poggibonsi (SI) - Tel. 0577/937336

Cambio, vendo **programmi** per C64: Legge 373, ingegneria, Word Processing, giochi, gestionali (fatture, magazzino, gestione medici, ecc.) e tanti altri. Per ricevere la lista scrivere a: Maurizio e Franco Bruno - Via Giorgio Bratti, 100 - 47023 Cesena (FO)

Cerco urgentemente il programma **Tot 13** per il Commodore 64. Posso ricambiare con programmi come Tool Kit, Simon's BASIC o molti giochi in linguaggio macchina. Solo zona Terni. Claudio La Rosa - Viale Trento, 46 - 05100 Terni (TR) - Tel. 0744/2844118

Vendo o cambio **cassetta con 20 giochi** per VIC 20 inespanso a L. 20.000. Se volete scambiare mandatemi la vostra lista che vi manderò la mia. Massima serietà. Annuncio sempre valido. Giovanni Bongiorni - Via Merosi, 11 - 29100 Piacenza (PC) - Tel. 0523/60475

Compro, cambio, vendo **programmi** per Commodore VIC 20. Ho una nastroteca di circa 120 programmi. Inviate le vostre liste e/o richieste a: Cosimo Tantillo - Via Luigi Rizzo, 15 - 90010 Aspra (PA) - Tel. 091/930314

Cambio **programmi** per CBM 64. Annuncio sempre valido. Rispondo a tutti, telefonare o scrivere a: Solaro Paolo - Piazza Medaglie d'Oro, 13 - 14100 Asti - Tel. 0141/51973

Vendo, cambio oltre **1.000 programmi** per Commodore 64. Marco Bombonato - Via N. Bixio, 27 - 20129 Milano - Tel. 02/224196

Per Commodore 64 vendo **giochi e utility** a prezzi favolosi, giochi a L. 5.000 massimo, utility a L. 20.000 massimo. Vendo a L. 50.000 20 giochi, a L. 100.000 45 giochi. Si omaggia di Turbo Tape per ordini di almeno tre giochi. Walter Mughini - Via Boccherini, 7 - 50144 Firenze - Tel. 055/367931

Compro, cambio, vendo **software** per CBM 64. Solo zona Catania. Fabio Bellassai - Via Fratelli Bandiera, 13 - 95100 Catania - Tel. 095/415353

Amici, se volete programmi per il vostro CBM 64 scrivetemi e potrete ricevere la lista, con prezzi davvero interessanti. Brambillasca Maria Luisa - Via Gramsci, 23/2 - 20041 Agrate Brianza (MI)

Per VIC 20 cambio o vendo **programmi in linguaggio macchina**. Ne possiedo più di 200, di cui alcuni molto belli. Fernando Benini - Via E. Pazzi, 16 - 48100 Ravenna

Compro, cambio, vendo **cassette con programmi** per Commodore 64. Compro giochi e programmi per la scuola ad un prezzo non superiore alle L. 25.000. Inviare cassetta e lettera o lista a: Massimo Dori - Via Dei Molini - 52037 Sansepolcro (AR)

Vendo per CBM 64 **centinaia di programmi di tutti i generi** a prezzi stracciati. Stefano Massoli - Via Massari, 10 - 06100 Perugia - Tel. 075/28983

Per C64 vendo, cambio **programmi di ogni genere** (games, utilities) su cassetta. I videogames sono in LM, i migliori del 1984 (Popeye, Decathlon, Zaxxon, ecc.). Annuncio sempre valido. Maurizio Borrelli - Via Firenze, 32 - 80100 Napoli - Tel. 081/281672

Cambio, vendo **software di tutti i generi** per Commodore 64. Elenco di circa 500 titoli tutti di buona qualità (98% in LM). Assicuro serietà e rapida risposta a tutti. Roberto Quaglia - Via Martinazzoli, 2 - 20161 Milano -Tel. 02/6462130

■ Cambio e vendo **oltre 1500 programmi** per Commodore 64. Per informazioni scrivere o telefonare a: Marcello Prudente - Via Luigi Tripoti, 7/A - 64100 Teramo - Tel. 0861/411184 ■

■ Per CBM 64 dispongo di **manuali tradotti in italiano e ultime novità** (Volo su Mosca, Fast Disk). Marcello Cesi - Via Magliana Nuova, 178 - 00146 Roma - Tel. 06/5266009 ■

Per CBM 64 cambio, vendo **circa 400 programmi** preferibilmente su disco. Roberto Manzardo - Via Rossini, 10 - 31029 Vittorio Veneto (TV) - Tel. 0438/560656

Cambio, vendo **diversi programmi** per C64: Miner, Burgertime, Mr. Robot. Dieci giochi L. 15.000, Turbo Tape L. 10.000, 30 giochi più Turbo Tape L. 50.000. Cambierei utilities; scrivete per la lista, massima serietà. Giancarlo Turbo - Via Ferrovia, 36 - 81057 Teano (CE) - Tel. 0823/875928

Vendo, ma sopratutto cambio, **programmi** per VIC 20. Ne possego molti in linguaggio macchina e BASIC, sia giochi che utility. Assicuro massima serietà, rivolgersi a: Michele e Nicola Giorato - Via M. Grappa, 11 - 35010 Cadoneghe (PD) - Tel. 049/704021

Vendo **programmi** per VIC 20 in versione base ed espanso a 3-8-16 Kbyte tra cui: Frogger (3 Kbyte), The Frog (versione base), Galaxian, QBert (16 Kbyte), ecc. Dispongo inoltre di programmi di utilità: Easyword, Maxischermo (8/16 Kbyte). Giuseppe Venezia - 10147 Torino - Tel. 011/210071

Compro, cambio, vendo **giochi** per C64: Zaxxon, Popeye, Dig Dug, International Soccer a L. 6.000 cadauno oppure 20 programmi a L. 53.000. Scrivere o telefonare dalle 20 alle 22 a: Daniele Noris - Via S. Bernardino, 1/A - 24100 Bergamo - Tel. 035/224500

Si è costituito il Commodore Computer Club per i possessori del CBM 64. Vendiamo giochi su disco e cassetta a L. 2.000 e L. 1.000. Risponderemo a tutti. Angelo Orlandi - Via Delle Albizzie, 40 - 00172 Roma - Tel. 06/288368

Per C64 cambio, vendo i programmi **Football, Totocalcio, Grand Prix** e un'infinità di bellissimi programmi su cassetta con listato. Giovanni Pugliese - Via A. Volta, 93 - 74100 Taranto - Tel. 099/413769

Cambio o vendo **giochi ed utility** per CBM 64 (prezzi estremamente bassi). Richiedere la lista con oltre 1.000 programmi scrivendo o telefonando a: Giovanni Melone - Viale Trieste, 33 - 81022 Casagiove (CE) - Tel. 0823/468497

Cambio, vendo **oltre 200 programmi su disco** per Commodore 64 a prezzi bassissimi. Richiedere la lista a: Riccardo Menchetti - Via A. Canova, 21 - 58100 Grosseto

■ Compro, cambio per Commodore 64 qualunque programma. **Dispongo di un archivio di oltre 1.500 programmi** e di traduzioni in italiano dei più famosi giochi (es. Flight Simulator II). Arrivi giornalieri dall'estero. Telefonare ore serali. Massimo Mattioz - Piazzale Accursio, 4 - 20100 Milano - Tel. 02/367373 ■

Desidero contattare appassionati di linguaggio macchina sul CBM 64 per scambio idee e informazioni. Dispongo di molto software e manuali. Sono interessato anche al drive 1541. Massimiliano Marras - Via Del Serafico, 64 - 00142 Roma - Tel. 06/5912597

Vendo **CBM 64 (un anno di vita)**, più registratore, 400 programmi a L. 750.000 trattabili. Gianluca Tarasconi - Via Properzio, 3 - 20135 Milano - Tel. 02/57410

■ Vendo **VIC 20,** registratore C2N, espansione 8 Kbyte, trislot, 3 cartucce, 7 libri. Regalo inoltre 11 cassette con molti giochi e utility in LM. Tutto in ottime condizioni a L. 250.000. Luca Dal Molin - Via G. Mazzini, 24 - 36030 Rettorgole (VI) - Tel. 0444/985673 ■

Cambio, vendo per Commodore 64 **oltre 1.000 programmi** a prezzi stracciati. Dispongo di utility e games. Telefonare o scrivere a: Giangualtiero Guidetti - Via Cigna, 86 - 10152 Torino - Tel. 011/288272

Per CBM 64 cambio **programmi, sia giochi che utilities.** Risposta assicurata a chiunque invia la propria lista a: Maria A. Monti - Casella Postale 45 - 55052 Fornaci di Barga (LU)

Per C64 compro, cambio, vendo **giochi e utility** a L. 3.000 l'uno tra cui Simon's BASIC, The Hobbit, Get Of My Garden, Decathlon, Baseball, One On One, Totocalcio. Acquisto minimo due programmi. Andrea Pecora - Via 2 Giugno, 16 - 52042 Camucia (AR) - Tel. 0575/62547

Occasione! Causa cambio computer, vendo **10 dischetti completamente registrati su entrambe le facciate** per C64, con più di 130 programmi nuovissimi. Massimo Proia - Via Pubblico Passeggio, 16 - 29100 Piacenza - Tel. 0523/32417

Vendo **programmi per geometra** per C64. Esempio: travi continue, teoria, errori, carnot, semi, nepero, ecc. Li preparo anche su richiesta, accompagnati da videogiochi a prezzi accessibilissimi. Michele Costanzo - Via Vittorio Emanuele, 386 - 90134 Palermo - Tel. 091/335175

Cambio, vendo per C64 **una cassetta contenente 6 programmi** in Turbo Tape (Pole Position, Zaxxon, Qix, Sam-Reciter, Donkey Kong, Juice) a sole L. 10.000 escluse spese di spedizione. Massima serietà. Luca Dell'Anna - Via Avellino, 12 - 73100 Lecce - Tel. 0832/591157

Vendo per VIC 20: Cabinet 6 slot a L. 200.000, super expander più 3 Kbyte a L. 35.000, sintetizzatore vocale L. 140.000, 16 Kbyte a L. 80.000 e VIC 20 a L. 110.000. Tutto in ottime condizioni. Telefonate dopo le 14 oppure scrivete a: Beppe De Vanna - Via Fontanelle II, 8 - 70057 Polese (BA) - Tel. 080/320427

Vendo **programmi** per VIC 20: cassetta con 70 programmi in versione base a L. 30.000 o cassetta con 25 programmi 8 Kbyte a L. 30.000. A scelta giochi o utility. Luciano Baglioni - Via Della Verna, 20 - 00141 Roma - Tel. 06/8185710

Compro, cambio, vendo **programmi** per CBM 64. Vorrei fondare un club. Matteo Oliviero - Via Marconi - 80056 Ercolano (NA) - Tel. 7393829

Vendo **circa 160 programmi** per C64 a sole L. 1.500 l'uno (minimo 10), a tutti regalo Turbo Tape. Qualche titolo: Hobbit, Zaxxon, BC, Buck Rogers, Pitfall, Decathlon, Beam Rider, 80 Colonne, Magic Desk, Koala Painter. Davide Rolando - Via B. Ottaviano, 6/6 - 17100 Savona - Tel. 019/26949

Cambio **oltre 300 programmi** per Commodore 64, sia utility che giochi. Richiedete e inviate la lista a: Cesare Boncompagni - Via Piave, 22 - 57013 Rosignano Solvay (LI) - Tel. 0586/76157

Cambio, vendo programmi per CBM 64. Dispongo di **oltre 500 programmi** che vendo a prezzi irrisori, specialmente in blocco. Specificare il supporto (nastro o disco). Annuncio sempre valido. Scrivere a: Alessandro Stoppaccioli - Via Vasco De Gama, 13 - 06034 Foligno (PG) - Tel. 0742/52413

Causa cessata attività svendo per Commodore 64 **Easy Script più manuale** a sole L. 35.000 (pagati L. 70.000). Roberto Bianchi - Via Ardigò, 1 - 20052 Monza (MI) - Tel. 039/360493

Al miglior offerente vendo **CBM 64** più floppy, monitor, datacassette e tutto il miglior software su disco. Materiale nuovo, con solo 4 mesi di vita e ancora in garanzia. Disponibile per qualunque prova. Luigi Roberto Callegari - Via Alcide De Gasperi, 47 - 21040 Sumirago (VA) - Tel. 0331/909183

PERSONAL SOFTWARE LIBRI — Gruppo Editoriale Jackson

Vuoi ordinare dei libri? Spedisci questo tagliando a:
Gruppo Editoriale Jackson
Via Rosellini, 12 – 20124 Milano

Nome Cognome

Indirizzo N.

Cap. Città Provincia

Partita I.V.A. (indispensabile per le aziende)

Si richiede l'emissione della fattura ☐

Inviatemi i seguenti libri:

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

☐ Pagherò al postino il prezzo indicato + L. 3.000 per contributo fisso spese di spedizione.

☐ Allego assegno n° .... di L. ....

☐ Non abbonato ☐ Abbonato sconto ☐ l'Elettronica ☐ Elettronica Oggi ☐ Automazione Oggi ☐ Elektor ☐ Informatica Oggi
☐ Computerworld ☐ Bit ☐ Personal Software ☐ Strumenti Musicali ☐ Video Giochi ☐ Home computer
☐ Telecomunicazioni Oggi ☐ PC Magazine ☐ Personal O ☐ Compuscuola ☐ Super VIC ☐ Super Sync

Data .... Firma ....

PERSONAL SOFTWARE PICCOLI ANNUNCI

Sei un lettore di Personal Software e vuoi entrare in contatto con tutti gli altri lettori per comprare, cambiare o vendere il tuo sistema o qualche particolare accessorio? Spedisci questo tagliando a:
Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

☐ COMPRO ☐ CAMBIO ☐ VENDO

☐ Annuncio gratuito ☐ Annuncio evidenziato (allegare L. 5.000). Non si accettano fotocopie o fogli volanti allegati al coupon.

Nome .... Cognome ....

Via .... N. .... C.A.P. ....

Città .... Prov. .... Tel. ....

UNA PUBBLICAZIONE
DEL GRUPPO EDITORIALE JACKSON

PERSONAL SOFTWARE

ANNO 4 N. 25 FEBBRAIO 1985

DIRETTORE RESPONSABILE: Giampietro Zanga

CAPO REDATTORE: Lorenzo Barrile

DIRETTORE TECNICO: Riccardo Paolillo

COORDINAMENTO TECNICO: Luca Zaninello

GRAFICA E IMPAGINAZIONE: Ugo Luzzati

SEGRETARIA DI REDAZIONE: Silvia Borgonovo

REDAZIONE USA: Gabriella Martino, Sergio Mello-Grand
G.E.J. Publishing Group Inc.

HANNO COLLABORATO A QUESTO NUMERO:
M. Mannuzzi, C. Tovena, M. Sangiorgio,
A. Motta, L. Lotti, M. Grizzi,
G. Bellegatti, G. Puccio, E. Bove,
S. Cerutti, G. Pisani, M. Giacobazzi,
M. Guida, S. Borsani, M. Spero

UFFICIO ABBONAMENTI: Tel. 6880951/2/3/4/5

USA: G.E.J. Publishing Group, Inc. a Subsidiary of -
Gruppo Editoriale Jackson 1307 S. Mary Avenue -
94087 Sunnyvale (CA) Tel. (408) 7730103 - Telex: 49959972

GRAN BRETAGNA: G.E.J. Publishing L.t.d. 187 Oxford Street
London WIR1AJ Tel. (01) 4392931 - Telex: (051) 21248

PUBBLICITA': Concessionario per l'Italia e l'Estero
j Advertising S.r.l. Via Restelli, 5 - 20124 Milano
Tel. (02) 6882895-6882458-6880606 Telex: 316213 REINA I

PUBBLICITA' PER USA E CANADA:
International Media Marketing 16704 Marquardt Avenue,
P.O. Box 1217 Cerritos (CA) 90701 - (213) 926-9552

FOTOCOMPOSIZIONE: Lineacomp S.r.l.
Via Rosellini, 12 - 20124 Milano

STAMPA: Litografica S.r.l. (Busto Arsizio - MI)

Concessionario esclusivo per la diffusione
in Italia e all'Estero: SODIP - Via Zuretti, 25 - 20125 Milano

AUTORIZZAZIONE ALLA PUBBLICAZIONE:
Tribunale di Milano n. 69 del 20/2/1982

Spedizione in abbonamento postale Gruppo III/70
Prezzo della rivista L. 4.000. Numero arretrato L. 8.000.
Abbonamento annuo (11 numeri) L. 34.000; per l'Estero L. 51.000
I versamenti vanno indirizzati a: Gruppo Editoriale Jackson
Via Rosellini, 12 - 20124 Milano - mediante emissione
di assegno bancario, cartolina vaglia o utilizzando
il C/C postale numero 11666203.
Per i cambi di indirizzo, indicare, oltre naturalmente al
nuovo, anche l'indirizzo precedente, ed allegare alla
comunicazione l'importo di L. 500, anche in francobolli.

Il Gruppo Editoriale Jackson è iscritto nel Registro nazionale
della Stampa al n. 117 - Vol. 2 - Foglio 129 in data 17-8-1982

GRUPPO EDITORIALE JACKSON S.r.l.

DIREZIONE, REDAZIONE, AMMINISTRAZIONE:
Via Rosellini, 12 - 20124 Milano - Telefoni: 68.80.951/2/3/4/5

SEDE LEGALE: Via G. Pozzone, 5 - 20121 Milano

DIREZIONE EDITORIALE: Daniele Comboni

DIREZIONE AMMINISTRATIVA: Giuliano Di Chiano

DIREZIONE DIVISIONE PERIODICI: Dario Tiengo

DIREZIONE DIVISIONE LIBRI
E GRANDI OPERE: Roberto Pancaldi



Libri firmati JACKSON



Gabriele Ugolini

PERSONAL GRAPHICS

Vengono presentati e descritti un buon numero di programmi di grafica e animazione su APPLE.
La trattazione inizia con i "mattoni" del disegno, punti, linee e superfici, prosegue con l'osservazione di figure nel piano e oggetti nello spazio e la raffigurazione realistica di complesse figure geometriche, sfiora la grafica del video gioco e si conclude con l'animazione stile Walt Disney.

158 pagine

Codice **555D L. 22.000**

Michel Benelfoul

METODI DI REALIZZAZIONE DEI PROGRAMMI

Destinato agli utenti di personal con una certa pratica di programmazione BASIC, il libro propone un metodo per la realizzazione dei programmi.
Per smitizzare il "dialetto" dell'informatica comincia col fornire un glossario completo e rigoroso.
È poi affrontato il problema dell'analisi di un sistema, con particolare attenzione al metodo di scelta dell'hardware, alle possibili riorganizzazioni del sistema ed alle esigenze di elaborazione dei dati.
Col supporto di modelli basati sulla matematica moderna e sulla teoria degli insiemi si affronta infine il problema dell'organizzazione logica dei dati.

96 pagine
Codice **401H L. 10.000**

Reinhold Thurner

PROGRAMMAZIONE STRUTTURATA
Corso di autoistruzione

Questo corso di autoistruzione insegna i principi fondamentali della programmazione strutturata, principi che sono comuni a tutte le effettive applicazioni di questa importante metodologia. Esso è concepito per aiutare il lettore a capire i costrutti ed a applicare correttamente le tecniche della programmazione strutturata. Spiega anche e insegna l'uso corretto delle principali tecniche di rappresentazione strutturata usata in analisi e programmazione (diagrammi di flusso, struttogrammi, pseudocodice e diagrammi ad albero).

136 pagine

Codice **503A L. 13.500**

La Biblioteca che fa testo

CEDOLA DI COMMISSIONE LIBRARIA

VOGLIATE SPEDIRMI

n° copie	codice	Prezzo unitario	Prezzo totale
		Totale	

☐ Pagherò contrassegno al postino il prezzo indicato più **L. 3000 per contributo fisso spese di spedizione.**

Condizioni di pagamento con esenzione del contributo spese di spedizione:

☐ Allego assegno della Banca

n° ____________

☐ Allego fotocopia del versamento su c/c n. 11666203 a voi intestato

☐ Allego fotocopia di versamento su vaglia postale a voi intestato

Nome
Cognome
Via
Cap Città Prov.
Data Firma

Spazio riservato alle Aziende. Si richiede l'emissione di fattura

Partita I.V.A.

ORDINE MINIMO L. 50.000

Attenzione compilare per intero la cedola
ritagliare (o fotocopiare) e spedire in busta chiusa a:
GRUPPO EDITORIALE JACKSON
Divisione Libri
Via Rosellini, 12 - 20124 Milano

GRUPPO EDITORIALE JACKSON

nuovidea

INCREDIBILE

TASTIERA - MIDI - SEQUENCER - BATTERIA PROGRAMMABILE
COMPUTER COMPATIBILE...



MK900
MIDI KEYBOARD

Tastiera portatile stereo - MIDI compatibile - a doppia generazione sonora

Possibilità di collegamento a computers



Supporto stand ripiegabile
(opzionale)

Una straordinaria ricchezza timbrica e una insuperabile versatilità sintetizzate in uno strumento a microprocessore dalla estrema facilità d'uso



Pedale d'espressione
(opzionale)

MIDI IN e MIDI OUT

10 ritmi + 1 ritmo programmabile dall'utente

10 Presets a doppia generazione sonora

Sequencer in tempo reale: 260 note + pause, 50 accordi, batteria per memorizzazione dati



Midi Computer Interface
(opzionale)

Divisione della tastiera programmabile che permette di suonare contemporaneamente 2 timbri oppure un solo timbro con polifonia 14

Demo Song

Accompagnamento automatico multifunzione

Controcanto automatico

Transpose, Detune, Stereo Chorus

Amplificazione stereo con due altoparlanti biconici a sospensione pneumatica incorporati

Tastiera a 61 tasti

Peso: kg. 6

SIEL®

Distribuito da
ARAMINI
STRUMENTI MUSICALI
Cadriano di Granarolo, via B. Buozzi, 1b (Bologna)
Tel. 051/766.077